Report No. NASAO/TRAINING/2011/1

Network for Applied Systems Analysis & Optimization

MATLAB/SCILAB for Scientific & Engineering Systems Analysis

being main lectures delivered at the course organised by
NETWORK FOR APPLIED SYSTEMS ANALYSIS & OPTIMIZATION
held at
Computing Centre
University of Nigeria, Nsukka, Nigeria

during the period
28 November–2 December 2011
edited by
Prof. S. O. Enibe

21 NOVEMBER 2011

List of Contributors

Prof. S. O. Enibe Samuel Ogbonna Enibe is a Professor of Mechanical Engineering, University of Nigeria, Nsukka. He was the Director of the National Centre for Equipment Maintenance and Development (NCEMD) from 1^{st} August 2008 to 18^{th} August 2011. He was Head, Department of Mechanical Engineering for several years.

Paul C. Oranu Mr. Paul C. Oranu has worked for several years as a core staff of the Management Information Systems Unit (MIS) of the University of Nigeria. He is currently the Acting Director of the Unit.

Prof. J. C. Agunwamba Professor J. Chukwuemeka Agunwamba is a Professor of Civil Engineering, University of Nigeria, Nsukka. He has served as Head, Department of Civil Engineering and became the Dean, Faculty of Engineering in August 2009.

Dr. Ernest Nweke Dr. Ernest Nweke is a Senior Lecturer in the Department of Mechanical Engineering, University of Nigeria, Nsukka. He worked for many years in the United States of America before assuming his present position in 2008.

W. I. E. Chukwu Walford I. E. Chukwu is an Associate Professor of Statistics with specialization in Operations Research. He has served as Head of Department of Statistics for several years.

Prof M. O. Oyesanya Professor M. O. Oyesanya is a Professor of Mathematics, University of Nigeria, Nsukka. After serving as Head of Mathematics for many years and the Associate Dean, Faculty of Physical Sciences, he became the substantive Dean in August 2009.

Prof. K. P. Baiyeri Professor Kayode Paul Baiyeri is a Professor of Crop Science, University of Nigeria, Nsukka. He served as Head of Crop Science prior to his present position as Dean, Faculty of Agriculture in 2009.

Prof. (Mrs.) P. O. Osadebe Professor (Mrs) Patience Ogoamaka Osadebe is a Professor of Pharmaceutical chemistry, University of Nigeria, Nsukka. She served for many years as Head, Department of Pharmaceutical Chemistry before her current appointment as Dean, Faculty of Pharmaceutical Sciences in 2009. She has also served as the National President of the Nigerian Association of Pharmacists in the Academia, (NAPA).

Rev. Dr. Ezichi A. Ituma Rev. Dr. Ezichi A. Ituma is a Senior Lecturer in the Department of Religion, University of Nigeria, Nsukka. He has served as Head of the Department and is currently the Coordinator of the Combined Social Sciences Programme. Rev. Dr. Ituma is an ordained Minister of Religion in the Presbyterian Church of Nigeria.

Prof. Emma Ezeani Professor Emmanuel Ezeani is a Professor of Political Science, University of Nigeria, Nsukka. He became the Dean, Faculty of the Social Sciences in August 2009.

Preface

This book is a compilation of the Lectures delivered by specially invited professionals drawn from differing disciplines at the special Trainng Course on **MATLAB/SCILAB for Scientific & Engineering Systems Analysis** held at the Computing Centre, University of Nigeria, Nsukka, Nigeria during the period 28^{th} November – 2^{nd} December 2011.

The training was organized to contribute towards meeting the increasing needs of researchers, scientists and engineers for extensive data analysis and visualization.

It is hoped that users will find the material presented very useful.

Samuel Ogbonna Enibe

Nsukka, Nigeria

November 23, 2011

Acknowledgements

Many people contributed towards making the production of this book possible. First, I must thank all of the contriutors who worked under much pressure to produce the draft copies of their lectures within the available time.

I also wish to thank all the persons who accepted to serve as Trainee Trainers and contributed their time and other resources to understand the inner working principles of MATLAB/SCILAB in order to train others during the technical sessions of the course. They are Ahia Chinedu C., Ogbuka Cosmas U, Engr Ikenna Agwubilo, Engr E. C. Okororigwe, Mr. Bassey Dominic Nkanang, Mr. Stanley O. Ugwoke, Engr Mkpamdi Eke, Engr. Howard Njoku, Mr. Patrick N. Okolo, Engr (Mrs) Mary Onyia and Engr Ifeanyichukwu N. Obeka.

Finally, I thank my wife, Ojiugo, our children and other members of the family who had to bear with several inconveniences as I worked on putting finishing touches to this book. May God bless you all.

Above all, I thank our God Almighty for His sustaining Grace and Power.

Samuel Ogbonna Enibe

Contents

1. Benefits of Open Source Software in Tertiary Education by Paul C. Oranu **1**
1.1 Preambles . . . 1
1.2 Open Source Software (OSS) . . . 1
1.2.1 Some Background . . . 1
1.2.2 What is open source software (OSS)? . . . 2
1.3 Why Open Source Software in Tertiary Education . . . 4
1.3.1 A food for thought: . . . 4
1.3.2 Making Educational Institutions to practice what they preach . . . 4
1.3.3 Other Benefits of Open source software in Tertiary Education . . . 4
1.3.4 Open Source Software vs Proprietary software . . . 5
1.3.5 License rights as against copyright laws (Escape vendor lock-in) . . . 5
1.3.6 Security . . . 5
1.3.7 Affordability or Cost . . . 5
1.3.8 Transparency . . . 5
1.3.9 Stability/Continuity . . . 6
1.3.10 Interoperability . . . 6

2. ENGINEERING APPLICATIONS OF NUMERICAL ANALYSIS by J. C. Agunwamba* & E. Nweke+ **8**
2.1 INTRODUCTION . . . 8
2.1.1 Definition . . . 8
2.1.2 Goals of Numerical Analysis . . . 9
2.1.3 History . . . 9
2.2 METHODS OF NUMERICAL ANALYSIS . . . 10
2.3 ELEMENTS OF NUMERICAL ANALYSIS . . . 11

2.3.1 Discretization . . . 11
2.3.2 The generation and propagation of errors . . . 11
2.3.3 Numerical stability and well-posed problems . . . 11
2.4 Areas of study . . . 12
2.4.1 Computing values of functions . . . 13
2.4.2 Interpolation, extrapolation, and regression . . . 13
2.4.3 Solving equations and systems of equations . . . 13
2.4.4 Solving eigenvalue or singular value problems . . . 13
2.4.5 Optimization . . . 14
2.4.6 Evaluating integrals . . . 14
2.4.7 Differential equations . . . 14
2.4.8 Software . . . 14
2.5 Application of Numerical Analysis . . . 15

3. Available software for data analysis and visualization in the Physical Sciences by Professor M. O. Oyesanya 18
3.1 Maple software . . . 19
3.2 MATHEMATICA SOFTWARE . . . 23
3.3 MATLAB (matrix laboratory) . . . 24
3.4 REPSN SOFTWARE: . . . 30
3.5 SINGULAR (typeset Singular) . . . 31
3.6 Conclusion . . . 31

4. Design of Experiments for Efficient Data Analysis/Visualization by Chukwu W. I. E 34
4.1 INTRODUCTION . . . 34
4.1.1 Introduction . . . 34
4.1.2 CHARACTERISTICS OF A WELL PLANNED EXPERIMENT . . . 36
4.1.3 1.3 PROCEDURE FOR EXPERIMENTATION . . . 36
4.1.4 SOME DEFINITIONS IN THE EXPERIMENT AND SURVEY DATA . . . 38
4.1.5 TOOLS FOR DEVELOPING OF EXPERIMENTAL DESIGN . . . 39
4.2 DIFFERENT TYPES OF DESIGNS APPLIED IN EXPERIMENTAL LAYOUTS . 40
4.2.1 COMPLETELY RANDOMIZED BLOCK DESIGN . . . 41
4.2.2 Factorial Completely randomized design . . . 43
4.2.3 LATIN SQUARE DESIGN . . . 47

4.2.4 Graeco Latin Square Design . . . 50
4.2.5 NESTED OR HIERARCHICAL DESIGN . . . 52
4.2.6 Split Plot Design . . . 58
4.2.7 How to Choose a Split-Plot Design . . . 59
4.3 Visualization . . . 61
4.3.1 What is it? . . . 62
4.3.2 Who's doing it? . . . 62
4.3.3 How does it work? . . . 63
4.3.4 Why is it significant? . . . 63
4.3.5 What are the downsides? . . . 63
4.3.6 Where is it going? . . . 64
4.3.7 What are the implications for teaching and learning? . . . 64

5. Commonly used software for data analysis and visualization in experimental agriculture by Baiyeri, K. P. 66
5.1 Introduction . . . 66
5.2 SAS (Statistical Analysis System) . . . 68
5.3 Statistical Package for Social Sciences (SPSS) . . . 69
5.4 GENSTAT . . . 70
5.5 MSTAT-C . . . 70
5.6 MS-EXCEL . . . 72

6. APPLICATION OF DATA ANALYSIS/DATA VISUALIZATION IN PHARMACEUTICAL RESEARCH AND DEVELOPMENT by OSADEBE, PATIENCE OGOAMAKA 75
6.1 Introduction . . . 75
6.1.1 Overview of Analytical statistical tools . . . 75
6.1.2 Definition of Terms . . . 76
6.1.3 Data types in pharmacy research . . . 76
6.2 Visualization methods . . . 77
6.2.1 Visualization tools for biological data . . . 77
6.2.2 Visualization tools for chemical data . . . 77
6.2.3 Data analysis in pharmaceutical research and development . . . 77
6.2.4 Visual molecular dynamics (VMD) . . . 78

6.2.5 PyMOL . . . 79
6.2.6 Chemcraft . . . 80
6.3 Conclusion . . . 81

7. Data Analysis/ Visualization in Social Sciences: Current Challenges and Future Trends by Ezichi A. Ituma* & Emma Ezeani+ 82
7.1 Introduction . . . 82
7.2 Social Sciences Methodologies . . . 83
7.3 Data Analysis . . . 84
7.3.1 Note on differences . . . 84
7.3.2 Qualitative Data Analysis . . . 84
7.3.3 Quantitative Data Analysis . . . 85
7.3.4 A word for SPSS . . . 85
7.4 Visualization . . . 86
7.5 Data Analysis/Visualization of Senate Paper on approval of Examination results . . . 86
7.5.1 Reality of the meaning of the data base on visualization . . . 87
7.5.2 Challenges . . . 88
7.5.3 Suggestions . . . 89
7.5.4 Trends . . . 89
7.6 Conclusion . . . 89

8. DATA/SYSTEMS ANALYSIS WITH MATLAB/SCILAB by S. O. ENIBE 95
8.1 Introduction . . . 95
8.2 Applications . . . 96
8.2.1 Basic functions . . . 96
8.2.2 Toolboxes . . . 97
8.3 Project Data Types . . . 100
8.3.1 Numerical Data . . . 100
8.3.2 String Data . . . 100
8.3.3 Signals/Stream Data . . . 101
8.3.4 Simulated Data . . . 103
8.4 Graphics . . . 103
8.4.1 Bar Charts . . . 104
8.4.2 Pie charts . . . 105

8.4.3 2D plots . . . 106
8.4.4 3D and other plots . . . 112
8.5 Input/output . . . 112
8.5.1 Input . . . 113
8.5.2 Output . . . 116
8.6 Statistical Analysis . . . 118
8.7 Recommended Procedure . . . 120
8.7.1 Data Generation . . . 120
8.7.2 Data Entry . . . 120
8.7.3 Data Cleaning . . . 121
8.7.4 Data Analysis/Plotting/Visualization . . . 121
8.8 Report Framework Generation . . . 121
8.8.1 Including Tables . . . 122
8.8.2 Including Figures . . . 123
8.8.3 Report Preparation . . . 123
8.9 Examples . . . 124
8.9.1 Examination Results . . . 124
8.9.2 Counting of Strings . . . 125
8.9.3 General Mathematical Functions . . . 126

INTRODUCTION TO SCILAB **134**

List of Tables

2.1 Example of Babylonian Method 12
4.1 ANALYSIS OF RESULTS OF AN EXPERIMENT IN WHICH COMPLETELY RANDOMIZED BLOCKS DESIGN IS APPLIED 42
4.2 Structure of design of completely randomized blocks 43
4.3 Analysis of variance of completely randomized blocks 44
4.4 Analysis of variance of Factorial completely randomized blocks 46
4.5 The result of ANOVA 47
4.6 ANOVA for Table: Analysis of variance of Latin Square 49
4.7 5x5 Graeco-Latin Square Design 51
4.8 Analysis of variance for Graeco-Latin Square Design 51
4.9 Tests of Between-Subjects Effects 53
4.10 54
4.11 Analysis of variance (ANOVA)for Nested Design 56
4.12 Analysis of Variance (Balanced Designs) 57
4.13 Source Variance Error Expected Mean Square 57
4.14 ANOVA Table for split-plot designs: 61
4.15 Tests of Between-Subjects Effects 62
7.2 Distribution of 2010/2011 Graduating Students' Results in the Faculty of Agriculture 86
7.4 Distribution of 2010/2011 Graduating Students' Results in the Faculty of the Social Sciences 87
7.1 Faculty of Social Sciences Unprocessed Results Chart 92
7.3 Faculty of Social Sciences Unprocessed Results Chart 93
7.5 Faculty of Agriculture Failure Chart 94
7.6 Faculty of Agriculture Unprocessed Results Chart 94
8.1 Available toolboxes in SCILAB 97
8.2 Signal processing functions in SCILAB 101

8.3 An example code to create a barchart 105
8.4 Code to create the piechart shown in figure 8.3 108
8.5 Sample Scilab code for 2-d plots of data 110
8.6 Sample Scilab code for 3-D plots of funstions 112
8.7 Sample Scilab code to open and read a Microsoft Excell file 115
8.8 Sample Scilab code to write data to a comma separated value file 117
8.9 Basic statistical functions available in SCILAB 118
8.10 Statistical toolboxes available in SCILAB 120
8.11 LaTeX sectioning depth values for use in in function *latexreport* 122
8.12 Example Grading System 124
8.13 Sample of responses on rating of equipment 127
8.14 An example code to count strings from a field survey 128
8.15 Rating/Performance of Equipment 129
8.16 Approximate latitude and longitude (in degrees) for some towns in Nigeria 129
8.17 Recommended Average Days for each Month and Values of n by Months 130
8.18 Daylength and sunset hour angle for selected Nigerian cities 131
8.19 Code to calculate the daylight hours 132

List of Figures

4.1 Illustration of nested design 54
5.1 A GGE biplot analysis visualizing five accessions of African Walnut in relation to their nutritional qualities 73
5.2 A GGE biplot analysis visualizing processing methods and ripening stages of plantain in relation to nutritional qualities of the fruit. 74
8.1 An example bar chart produced with code and data in table 8.3 106
8.2 A simple pie chart 107
8.3 A pie chart with legend placed outside 109
8.4 Number of hours of daylight at Abuja, Nsukka and Sokoto 111
8.5 Illustratrations in 3D produced with the Scilab codes shown in table 8.6 113
8.6 Pie chart illustrating the responses on equipment rating 129

MATLAB/SCILAB Training Course, Vol. 1, pp. 1–7, 2011

Received 02 November 2011

CHAPTER 1

BENEFITS OF OPEN SOURCE SOFTWARE IN TERTIARY EDUCATION

by

PAUL C. ORANU

Ag. Director, MIS Unit, University of Nigeria, Nsukka

1.1 Preambles

Open source describes a practice in production and development in which the raw materials of an end product is made available to the users of the end product. In most cases, open source actually promotes access to the source material allowing input and modifications. It should be noted that the phrase "open source" is not exclusively associated with software. In fact, the phrase actually predated computer and indeed other products have their own open source versions, for instance, beverages (eg OpenCoke an open source alternative to Coca cola), electronics (eg openmoko – an open source mobile phone), health sciences, etc.

1.2 Open Source Software (OSS)

1.2.1 Some Background

Initially, software were free (libre software). When IBM and others sold the first large-scale commercial computers, in the 1960s, they came with some software which were free (libre), in the sense that

they could be freely shared among users. They came with source code, and they could be modified and improved upon. Shortly after this however, proprietary software came in and gradually took over. Proprietary software actually started gaining ground around mid 1970s. They were proprietary in the sense that users were not allowed to redistribute them, that source codes were not available, and that users could not modify the programs.

From late 1970s and early 1980s, various individuals and groups started different initiatives that ended up in what we today call open source software. Notable among these were Richard Stallman, formerly a programmer at the MIT AI Lab, who resigned, and launched the GNU Project, and Computer Science Research Group (CSRG) of the University of California at Berkeley who were then working on the Unix system to come up with the open source version of UNIX called BSD. It was however in Netscape's January 1998 announcement of a source code release for Navigator that a group of individuals at the session that included Christine Peterson suggested “open source”, as the name instead of free software.

1.2.2 What is open source software (OSS)?

- Open Source Software is therefore software that has been released under a license which allows access to the source code, permits users to study, change, improve and at times also distribute the software.

Here are some examples of OSS:

- Moodle (virtual learning system)
- OpenOffice.org (office suite, including word processor, spreadsheet, and presentation software)
- GNU
- BSD, BSD2,
- Apache HTTP Server (web server)
- Blender (3D graphics and animation package)
- DSpace (digital repository)
- EPrints (digital repository)
- The GIMP (image editor),
- GNOME (Linux desktop environment)

- GNU Compiler Collection (GCC, a suite of compilation tools for C, C++, etc)
- KDE (Linux desktop environment)
- LORLS (reading lists management system)
- Mailman (mailing list manager)
- Mozilla] (web browser and email client)
- Firefox] (web browser based on Mozilla)
- Thunderbird (mail client based on Mozilla code)
- MySQL (database)
- PHP (web development)
- Perl (programming/scripting language)
- Plone (content management system)
- PostgreSQL (database)
- Python (programming/scripting language)
- Sakai (learning management system)
- Samba (file and print server)
- SSL-Explorer: Community Edition (browser-based SSL VPN solution)
- TEX /LATEX (typesetting language)
- WUBS (resource booking system)
- XFree86 (implementation of the X window system)
- Zope (web application server)
- Etc.

It is clear from these examples that the majority of the OSS software being used are in server infrastructure with fewer being used on normal PCs.

1.3 Why Open Source Software in Tertiary Education

1.3.1 A food for thought:

It is indeed a strange world when educators need to be convinced that sharing information, as opposed to concealing information, is a good thing. The advances in all of the arts and sciences, indeed the sum total of human knowledge, are the result of the open sharing of ideas, theories, studies and research. Yet throughout many school systems, the software in use on computers is closed and locked, making educators partners in the censorship of the foundational information of this new age.

1.3.2 Making Educational Institutions to practice what they preach

Working on Open source software encourages students to learn by practicing with real life situations and dealing with real world problems. They learn by working on actual software codes in use and see the effect of their learning practices. It also uses the power of collaboration to provide students with hands-on learning and to equip students with an expanded skill set that is very attractive to businesses. This in particularly important as they will work in a matrix environment where projects cut across organizational and geographic boundaries, requiring cooperation and communication when they graduate.

Teaching students to learn by collaboration and networking early in their lives is very important as this makes them come out as ready team players and better prepares students for future job experiences. This way, Open-source developers don't rely on textbooks; they rely on the knowledge base of other developers with whom they connect through community forums, building off of one another's ideas to create a solution that is eventually shared with all. Students that learn with a lot of open source software experience gain skills in collaboration, project management, and testing and encourages a well-rounded computer science education, making students more marketable in the business world.

1.3.3 Other Benefits of Open source software in Tertiary Education

Considerable interest has been shown in OSS by tertiary education institutions because of the promise of a reduced total cost of ownership of the software, potentially better support, freedom from vendor hijack or dependence also know as vendor lock-in, ability to customize the software and opportunity to contribute to knowledge and learning. Put in other words, there are pedagogic benefits of being able to view and modify the source code, and learn from the exercise. Of all these, it has been discovered that the major reasons for the increasing use of OSS in Tertiary Education is lower Total Cost of Ownership and freedom from software vendor dependence or vendor lock-in.

1.3.4 Open Source Software vs Proprietary software

Many reasons readily come to our mind why we should use OSS rather than proprietary software. We are however going pointed out a few of these reasons for adoption of open source here, in particular, the heightened value proposition from open source (when compared to most proprietary formats):

1.3.5 License rights as against copyright laws (Escape vendor lock-in)

Software licenses grant rights to users which would otherwise be reserved by copyright law to the copyright holder. Several open source software licenses have qualified within the boundaries of the Open Source Definition. The most prominent and popular example is the GNU General Public License (GPL). While open source distribution presents a way to make the source code of a product publicly accessible, the open source licenses allow the authors to fine tune such access.

1.3.6 Security

• Privacy, data protection

• Vulnerability to viruses

Most of us must have noticed the spade of virus attack on most proprietary software such as Microsoft products. The same cannot be said of OSS software like Linux, OpenOffice.org office suite, etc. Open source software is available publicly. A large amount of developers globally contribute and analyze the code making it more secure and constantly increasing the quality. The peer review process drive excellence in design.

1.3.7 Affordability or Cost

OSS are obtained at little or no cost unlike proprietary software where you have license renewal to contend with for ever. The issue of piracy flies out of the window with OSS around. Also, OSS will not require hardware upgrade or at least will not force hardware upgrade. Most Open source software are able to run effectively on older/ lowermost hardware (backward compatibility).

1.3.8 Transparency

People are free to make contribution, everything is open and all the cards are face up on the table. Due to the fact that OSS is available publicly, a large amount of skilled developers globally contribute and analyze the code and are constantly increasing the quality. This openness also encourages peer review, raising the bar for every contributor since each knows that his/her contribution will be reviewed worldwide. There is no possibility of a ‘black box’.

1.3.9 Stability/Continuity

Once you have obtain any OSS, how far you go with it is largely up to you and you own it forever, no license! Unless the job changes or more efficient processes are discovered then there is rarely pressure or need to alter the software that is being used to assist the task. This more or less run directly counter to what motivates software vendors who are in the unenviable position of supplying a commodity that does not wear out or age much. The vendors need a stable revenue stream to be able to keep their business going while their customers have not the slightest desire to change or upgrade any product that is working well enough to suit their needs. If a software supplier can establish a virtual monopoly and then force upgrades onto its audience (as has been the history of the software industry since the mid 1960s) then the profits can be very high.

It is possible that a proprietary software company closes shop or decides to discontinue services for a software. With it goes the support and the future of the software. Open source software is not dependent on a single entity and gives users the advantage of the community. Any developer can choose to take up the software and continue from where it stopped or others stopped.

1.3.10 Interoperability

Of course, ability to customize makes the product very portable and "environment free"

References

1. http://en.wikipedia.org/wiki/Open-source_software
2. http://open-source.gbdirect.co.uk/migration/benefit.html
3. http://www.cynapse.com/resources/benefits-open-source
4. (http://edge-op.org/grouch/schools.html)
5. David M. Berry (2008). Copy, Rip, Burn: The Politics of Copyleft and Open Source. London:Pluto Press. http://www.amazon.com/Copy-Rip-Burn-Politics-Source/dp/0745324142.
6. Karl Fogel. Producing Open Source Software (How to Run a Successful Free Software Project). Free PDF version available.
7. Ron Goldman and Richard P. Gabriel (2005). Innovation Happens Elsewhere. Richard P. Gabriel. ISBN 1558608893. http://dreamsongs.com./IHE/IHE.html

8. http://www.educause.edu/EDUCAUSE+Quarterly/EDUCAUSEQuarterly MagazineVolum/OpenSourceSoftwareinEducation/162873

9. Gary Hepburn and Jan Buley : Getting Open Source Software into Schools: Strategies and challenges http://www.mendeley.com/research/getting-open-source-software-schools-strategies-challenges/

MATLAB/SCILAB Training Course, Vol. 1, pp. 8–17, 2011

Received 4 May 2011

CHAPTER 2

ENGINEERING APPLICATIONS OF NUMERICAL ANALYSIS

by

J. C. AGUNWAMBA* & E. NWEKE+

***Department of Civil Engineering**

+Department of Mechanical Engineering

University of Nigeria, Nsukka

2.1 INTRODUCTION

2.1.1 Definition

Numerical analysis is the study of algorithms that use numerical approximation (as opposed to general symbolic manipulations) for the problems of mathematical analysis (as distinguished from discrete mathematics). Some use the term scientific computing to represent numerical analysis. However, the use of this term is criticised because one does not have to do "computing" to do numerical analysis. The words are interchangeable to a degree, but numerical analysis refers more to the theoretical side and scientific computing more to the practical side even though there is a large overlap. Some people indeed use scientific computing in a broad sense to include all of numerical analysis, but other people use numerical analysis to include all of scientific computing.

Other definitions of numerical analysis exist such as “the study of algorithms for the problems of continuous mathematics (as distinguished from discrete mathematics) that do not have an explicit, direct solution". Some mathematicians have argued that this is inaccurate. For instance, Gaussian

elimination is one of the most prominent algorithms in numerical analysis, yet it has nothing to do with continuous mathematics and the solution is direct. The definition includes only methods which suffer from truncation error and neglects those which suffer from only round-off error. This definition is certainly not supposed to include only methods suffering from truncation error. Gaussian elimination solves Ax = b where x is a vector of reals. This is a problem in continuous mathematics (x is chosen from a continuum) and not in discrete mathematics. The phrase "that do not have an explicit, direct solution" serves to distinguish numerical analysis from symbolic mathematics. However, the word "direct" is ill-chosen since Gaussian elimination is called a "direct method" as opposed to an "iterative method" like Gauss-Seidel. Another definition is that numerical analysis is the study of algorithms for obtaining approximations to the exact (but usually unknown) solutions of problems in continuous mathematics.

2.1.2 Goals of Numerical Analysis

The overall goal of the field of numerical analysis is the design and analysis of techniques to give approximate but accurate solutions to hard problems, the variety of which is suggested by the following.

- Advanced numerical methods are essential in making numerical weather prediction feasible.
- Computing the trajectory of a spacecraft requires the accurate numerical solution of a system of ordinary differential equations.
- Car companies can improve the crash safety of their vehicles by using computer simulations of car crashes. Such simulations essentially consist of solving partial differential equations numerically.
- Hedge funds (private investment funds) use tools from all fields of numerical analysis to calculate the value of stocks and derivatives more precisely than other market participants.
- Airlines use sophisticated optimization algorithms to decide ticket prices, airplane and crew assignments and fuel needs. This field is also called operations research.
- Insurance companies use numerical programs for actuarial analysis.

2.1.3 History

The field of numerical analysis predates the invention of modern computers by many centuries. Linear interpolation was already in use more than 2000 years ago. Many great mathematicians of the past were preoccupied by numerical analysis, as is obvious from the names of important algorithms like Newton's method, Lagrange interpolation polynomial, Gaussian elimination, or Euler's method.

To facilitate computations by hand, large books were produced with formulas and tables of data such as interpolation points and function coefficients. Using these tables, often calculated out to 16 decimal places or more for some functions, one could look up values to plug into the formulas given and achieve very good numerical estimates of some functions. The canonical work in the field is the NIST publication edited by Abramowitz and Stegun, a 1000-plus page book of a very large number of commonly used formulas and functions and their values at many points. The function values are no longer very useful when a computer is available, but the large listing of formulas can still be very handy.

The mechanical calculator was also developed as a tool for hand computation. These calculators evolved into electronic computers in the 1940s, and it was then found that these computers were also useful for administrative purposes. But the invention of the computer also influenced the field of numerical analysis, since now longer and more complicated calculations could be done.

2.2 METHODS OF NUMERICAL ANALYSIS

There are the direct and the iterative methods.

The direct methods compute the solution to a problem in a finite number of steps. These methods would give the precise answer if they were performed in infinite precision arithmetic. Examples include Gaussian elimination, the QR factorization method for solving systems of linear equations, and the simplex method of linear programming. In practice, finite precision is used and the result is an approximation of the true solution (assuming stability).

In contrast to direct methods, iterative methods are not expected to terminate in a number of steps. Starting from an initial guess, iterative methods form successive approximations that converge to the exact solution only in the limit. A convergence test is specified in order to decide when a sufficiently accurate solution has (hopefully) been found. Even using infinite precision arithmetic these methods would not reach the solution within a finite number of steps (in general). Examples include Newton's method, the bisection method, and Jacobi iteration. In computational matrix algebra, iterative methods are generally needed for large problems.

Iterative methods are more common than direct methods in numerical analysis. Some methods are direct in principle but are usually used as though they were not, e.g. GMRES and the conjugate gradient method. For these methods the number of steps needed to obtain the exact solution is so large that an approximation is accepted in the same manner as for an iterative method.

2.3 ELEMENTS OF NUMERICAL ANALYSIS

2.3.1 Discretization

Furthermore, continuous problems must sometimes be replaced by a discrete problem whose solution is known to approximate that of the continuous problem; this process is called discretization. For example, the solution of a differential equation is a function. This function must be represented by a finite amount of data, for instance by its value at a finite number of points at its domain, even though this domain is a continuum.

2.3.2 The generation and propagation of errors

The study of errors forms an important part of numerical analysis. There are several ways in which error can be introduced in the solution of the problem.

Round-off. Round-off errors arise because it is impossible to represent all real numbers exactly on a machine with finite memory (which is what all practical digital computers are).

Truncation and discretization errors. Truncation errors are committed when an iterative method is terminated or a mathematical procedure is approximated, and the approximate solution differs from the exact solution. Similarly, discretization induces a discretization error because the solution of the discrete problem does not coincide with the solution of the continuous problem. For instance, in the iteration in the sidebar to compute the solution of 3x3 + 4 = 28, after 10 or so iterations, we conclude that the root is roughly 1.99 (for example). We therefore have a truncation error of 0.01.

Once an error is generated, it will generally propagate through the calculation. For instance, we have already noted that the operation + on a calculator (or a computer) is inexact. It follows that a calculation of the type a+b+c+d+e is even more inexact.

What does it mean when we say that the truncation error is created when we approximate a mathematical procedure. We know that to integrate a function exactly requires one to find the sum of infinite trapezoids. But numerically one can find the sum of only finite trapezoids, and hence the approximation of the mathematical procedure. Similarly, to differentiate a function, the differential element approaches to zero but numerically we can only choose a finite value of the differential element.

2.3.3 Numerical stability and well-posed problems

Numerical stability is an important notion in numerical analysis. An algorithm is called numerically stable if an error, whatever its cause, does not grow to be much larger during the calculation. This happens if the problem is well-conditioned, meaning that the solution changes by only a small amount if the problem data are changed by a small amount. To the contrary, if a problem is ill-conditioned, then any small error in the data will grow to be a large error.

Both the original problem and the algorithm used to solve that problem can be well-conditioned and/or ill-conditioned, and any combination is possible.

So an algorithm that solves a well-conditioned problem may be either numerically stable or numerically unstable. An art of numerical analysis is to find a stable algorithm for solving a well-posed mathematical problem. For instance, computing the square root of 2 (which is roughly 1.41421) is a well-posed problem. Many algorithms solve this problem by starting with an initial approximation x1 to square root of 2, for instance x1=1.4, and then computing improved guesses x2, x3, etc.. One such method is the famous Babylonian method, which is given by xk+1 = xk/2 + 1/xk. Another iteration, which we will call Method X, is given by xk + 1 = (xk2−2)2 + xk.[3] We have calculated a few iterations of each scheme in table form below, with initial guesses x1 = 1.4 and x1 = 1.42.

Table 2.1: Example of Babylonian Method

Babylonian	Babylonian	Method X	Method X
$x_1 = 1.4$	$x_1 = 1.42$	$x_1 = 1.4$	$x_1 = 1.42$
$x_2 = 1.4142857...$	$x_2 = 1.41422535...$	$x_2 = 1.4016$	$x_2 = 1.42026896$
$x_3 = 1.414213564...$	$x_3 = 1.41421356242...$	$x_3 = 1.4028614...$	$x_3 = 1.42056...$
		...	...
		$x_{1000000} = 1.41421...$	$x_{28} = 7280.2284...$

Observe that the Babylonian method converges fast regardless of the initial guess, whereas Method X converges extremely slowly with initial guess 1.4 and diverges for initial guess 1.42. Hence, the Babylonian method is numerically stable, while Method X is numerically unstable.

Numerical stability is affected by the number of the significant digits the machine keeps on, if we use a machine that keeps on the first four floating-point digits,a good example on loss of significance these two equivalent functions if we compare the results of and by looking to the two above results, we realize that loss of significance which is also called Subtractive Cancelation has a huge effect on the results, even though both functions are equivalent; to show that they are equivalent simply we need to start by f(x) and end with g(x), and so the true value for the result is 11.174755...which is exactly g(500) = 11.1748 after rounding the result to 4 decimal digits now imagine that you tens of terms like these functions are used in the program; the error will increase as one proceeds in the program, unless one uses the suitable formula of the two functions each time one evaluates either f(x), or g(x); the choice is dependent on the parity of x.

2.4 Areas of study

The field of numerical analysis is divided into different disciplines according to the problem that is to be solved.

2.4.1 Computing values of functions

One of the simplest problems is the evaluation of a function at a given point. The most straightforward approach, of just plugging in the number in the formula is sometimes not very efficient. For polynomials, a better approach is using the Horner scheme, since it reduces the necessary number of multiplications and additions. Generally, it is important to estimate and control round-off errors arising from the use of floating point arithmetic.

2.4.2 Interpolation, extrapolation, and regression

Interpolation solves the following problem: given the value of some unknown function at a number of points, what value does that function have at some other point between the given points?

Extrapolation is very similar to interpolation, except that now we want to find the value of the unknown function at a point which is outside the given points.

Regression is also similar, but it takes into account that the data is imprecise. Given some points, and a measurement of the value of some function at these points (with an error), we want to determine the unknown function. The least squares-method is one popular way to achieve this.

2.4.3 Solving equations and systems of equations

Another fundamental problem is computing the solution of some given equation. Two cases are commonly distinguished, depending on whether the equation is linear or not. For instance, the equation 2x + 5 = 3 is linear while 2x2 + 5 = 3 is not.

Much effort has been put in the development of methods for solving systems of linear equations. Standard direct methods, i.e., methods that use some matrix decomposition are Gaussian elimination, LU decomposition, Cholesky decomposition for symmetric (or hermitian) and positive-definite matrix, and QR decomposition for non-square matrices. Iterative methods such as the Jacobi method, Gauss–Seidel method, successive over-relaxation and conjugate gradient method are usually preferred for large systems.

Root-finding algorithms are used to solve nonlinear equations (they are so named since a root of a function is an argument for which the function yields zero). If the function is differentiable and the derivative is known, then Newton's method is a popular choice. Linearization is another technique for solving nonlinear equations.

2.4.4 Solving eigenvalue or singular value problems

Several important problems can be phrased in terms of eigenvalue decompositions or singular value decompositions. For instance, the spectral image compression algorithm is based on the singular value decomposition. The corresponding tool in statistics is called principal component analysis.

2.4.5 Optimization

Optimization problems ask for the point at which a given function is maximized (or minimized). Often, the point also has to satisfy some constraints.

The field of optimization is further split in several subfields, depending on the form of the objective function and the constraint. For instance, linear programming deals with the case that both the objective function and the constraints are linear. A famous method in linear programming is the simplex method.

The method of Lagrange multipliers can be used to reduce optimization problems with constraints to unconstrained optimization problems.

2.4.6 Evaluating integrals

Numerical integration, in some instances also known as numerical quadrature, asks for the value of a definite integral. Popular methods use one of the Newton–Cotes formulas (like the midpoint rule or Simpson's rule) or Gaussian quadrature. These methods rely on a "divide and conquer" strategy, whereby an integral on a relatively large set is broken down into integrals on smaller sets. In higher dimensions, where these methods become prohibitively expensive in terms of computational effort, one may use Monte Carlo or quasi-Monte Carlo methods (see Monte Carlo integration), or, in modestly large dimensions, the method of sparse grids.

2.4.7 Differential equations

Numerical ordinary differential equations and Numerical partial differential equations. Numerical analysis is also concerned with computing (in an approximate way) the solution of differential equations, both ordinary differential equations and partial differential equations.

Partial differential equations are solved by first discretizing the equation, bringing it into a finite-dimensional subspace. This can be done by a finite element method, a finite difference method, or (particularly in engineering) a finite volume method. The theoretical justification of these methods often involves theorems from functional analysis. This reduces the problem to the solution of an algebraic equation.

2.4.8 Software

List of numerical analysis software and Comparison of numerical analysis software.

Since the late twentieth century, most algorithms are implemented in a variety of programming languages. The Netlib repository contains various collections of software routines for numerical problems, mostly in FORTRAN and C. Commercial products implementing many different numerical algorithms include the IMSL and NAG libraries; a free alternative is the GNU Scientific Library.

There are several popular numerical computing applications such as MATLAB, S-PLUS, LabVIEW, and IDL as well as free and open source alternatives such as FreeMat, Scilab, GNU Octave (similar to Matlab), IT++ (a C++ library), R (similar to S-PLUS) and certain variants of Python. Performance varies widely: while vector and matrix operations are usually fast, scalar loops may vary in speed by more than an order of magnitude.

Many computer algebra systems such as Mathematica also benefit from the availability of arbitrary precision arithmetic which can provide more accurate results. Also, any spreadsheet software can be used to solve simple problems relating to numerical analysis.

2.5 Application of Numerical Analysis

Numerical analysis is a very important tool in modelling. A typical objective of mathematical modelling is the derivation of analytical results which are generally valid in different situations. However, this is possible only for idealized problems, whereas for the majority of real world problems no analytical solutions exist and approximated results are only available. On the other hand, even a closed form solution is of little direct use to practitioners who need numerical values. The need to evaluate complicated formulae gave rise to the habit of tabulating functions and producing nomograms that fill a large volume of literature. The advent of powerful microcomputers has led to more and more application of numerical analysis in solving complex problems in engineering. Serious problems of engineering research that in the recent past were solvable only by using large mainframe units in computer centres with carefully controlled environments can be solved nowadays by inexpensive, powerful and robust microcomputers located on a desk of an individual scientist, engineer or student. Before the era of digital computers, nonlinear partial differential equations formulated for real world problems were of academic meaning only. Nowadays it is practical to solve these models and to use them in numerous real world applications using numerical analysis.

However, there are also negative bye products of advances in computer facilities. There exist several publications where results of academic importance are obtained in an overkill fashion through intensive computational effort, although they could have been developed analytically. In addition, there has being generation of many models, many of which do not improve on the understanding or the practical benefits obtained from existing models.

One of the earliest mathematical writings is the Babylonian tablet BC 7289, which gives a numerical approximation of square root of 2, the length of the diagonal in a unit square. Being able to compute the sides of a triangle (and hence, being able to compute square roots) is extremely important, for instance, in carpentry and construction. Numerical analysis continues this long tradition of practical mathematical calculations. Much like the Babylonian approximation of the square root of 2, modern numerical analysis does not seek exact answers, because exact answers are often impossible to obtain in practice. Instead, much of numerical analysis is concerned with obtaining approximate solutions while maintaining reasonable bounds on errors.

Numerical analysis naturally finds applications in all fields of engineering and the physical sciences, but in the 21st century, the life sciences and even the arts have adopted elements of scientific computations. Ordinary differential equations appear in the movement of heavenly bodies (planets, stars and galaxies); optimization occurs in portfolio management; numerical linear algebra is important for data analysis; stochastic differential equations and Markov chains are essential in simulating living cells for medicine and biology.

Before the advent of modern computers numerical methods often depended on hand interpolation in large printed tables. Since the mid 20th century, computers calculate the required functions instead. The interpolation algorithms nevertheless may be used as part of the software for solving differential equations. Numerical analysis is needed to solve engineering problems that lead to equations that cannot be solved analytically with simple formulas. Examples are solutions of large systems of algebraic equations, evaluation of integrals, and solution of differential equations. The finite element method is a numerical method that is in widespread use to solve partial differential equations in a variety of engineering fields including stress analysis, fluid dynamics, heat transfer, and electro-magnetic fields. The digital computer has enabled the wide use of numerical analysis in the last 50 years. But one must be cautious about numerical results due to possibilities of using the wrong theory and the "garbage in - garbage out"Comment

REFERENCES

1. Gilat, Amos (2004). MATLAB: An Introduction with Applications (2nd edition ed.). John Wiley & Sons. ISBN 0-471-69420-7.

2. Hildebrand, F. B. (1974). Introduction to Numerical Analysis (2nd edition ed.). McGraw-Hill. ISBN 0-070-28761-9.

3. Leader, Jeffery J. (2004). Numerical Analysis and Scientific Computation. Addison Wesley. ISBN 0-201-73499-0.

4. Trefethen, Lloyd N. (2006). "Numerical analysis", 20 pages. In: Timothy Gowers and June Barrow-Green (editors), Princeton Companion of Mathematics, Princeton University Press.

5. Numerische Mathematik, volumes 1-66, Springer, 1959-1994 (searchable; pages are images). (English) (German)

6. Numerische Mathematik at SpringerLink, volumes 1-112, Springer, 1959–2009

7. SIAM Journal on Numerical Analysis, volumes 1-47, SIAM, 1964–2009

8. First Steps in Numerical Analysis, R.J.Hosking, S.Joe, D.C.Joyce, and J.C.Turner

9. Numerical Analysis for Engineering, D. W. Harder

10. CSEP (Computational Science Education Project), U.S. Department of Energy

MATLAB/SCILAB Training Course, Vol. 1, pp. 18–33, 2011

Received 2 May 2011

CHAPTER 3

AVAILABLE SOFTWARE FOR DATA ANALYSIS AND VISUALIZATION IN THE PHYSICAL SCIENCES

by

PROFESSOR M. O. OYESANYA
Department of Mathematics
Faculty of Physical Sciences
University of Nigeria, Nsukka.

Man is driven by the urge to maximize all potentials around him and minimize time constraints. It is this urge that led him to bringing forth the programming languages that have evolved over the years and are still evolving. We want quick results that are dependable and leading us to useful conclusions. It's just like saying: Guys time is not on our side.

Some years ago in the 1950s going to England was by sea (in ships) and it takes weeks/months to get there. Then came the airplane and in about six hours (less than one day) you are in England. You can see the difference.

In the Physical Sciences a lot of computational analysis is done involving solution of equations (algebraic, differential, integral, integro-differential, difference, etc) linear and nonlinear of various complexities. There is therefore need for processes that will enhance dependable solutions within some reasonable time. There are some problems for which analytic methods cannot easily bring out. The resort is for approximate numerical analysis. This brings the need for algorithms that can be inexpensive but very reliable. This in turn brings in computer languages – Fortran, Pascal, Basic, C, C ++ etc. Writing programmes in these languages are sometimes not so simple. The process of simplifying the processes and minimizing time led to the coming of software packages.

In the Physical Sciences some software for data analysis and visualization are available. These include Maple, Mathematica, MATLAB, Scilab, Simulink Repsn and more.

3.1 Maple software

The first concept of Maple arose from a meeting in November 1980 at the University of Waterloo. Researchers at the university wished to purchase a computer powerful enough to run Macsyma. Instead, it was decided that they would develop their own computer algebra system that would be able to run on more reasonably priced computers.

The initial development of Maple proceeded very quickly, with the first limited version appearing in December 1980. Researchers tried and discarded many different ideas creating a continually evolving system. Maple was demonstrated first at conferences beginning in 1982.

In 1988 Waterloo Maple Inc. was founded. The company's original goal was to manage the distribution of the software. Eventually, the company evolved to have an R&D department where much of Maple's development is done today. Significant development of Maple continues at university research labs including: the Symbolic Computation Laboratory at the University of Waterloo; the Ontario Research Centre for Computer Algebra at the University of Western Ontario; and labs at other universities worldwide. The developers of Maple pioneered a number of significant algorithms emulated by other systems. These include breakthroughs in symbolic integration (e.g. see incomplete Gamma function and references herein), and many other functionalities.

In 1989, the first graphical user interface for Maple was developed and included with version 4.3 for the Macintosh. Prior versions of Maple included only a command line interface with two dimensional output. X11 and Windows versions of the new interface followed in 1990 with Maple V. Maple was used in a number of notable applications in Science and Mathematics ranging from a demonstration of Fermat's Last Theorem (proved by Chike Obi in 1999) in number theory, to solutions in General Relativity and quantum mechanics.

In 2003, the current "standard" interface was introduced with Maple 9 in 2003. This interface is primarily written in Java (although portions, such as the rules for typesetting mathematical formulae, are written in the Maple language). The Java interface was criticized for being slow; improvements have been made in later versions, although the Maple 11 documentation recommends the previous ("classic") interface for users with less than 500 MB of physical memory. This classic interface is no longer being maintained.

In 2008, Maple 12 added additional user interface features for making Maple easier to use as a MATLAB toolbox. The Core functionality of Maple software is that users can enter mathematics in traditional mathematical notation. Custom user interfaces can also be created. There is extensive support for numeric computations, to arbitrary precision, as well as symbolic computation and visualization. Examples of symbolic computations are given below.

Maple incorporates an imperative-style programming language which resembles Pascal. The language permits variables of lexical scope. There are also interfaces to other languages (C, C++, Fortran, Java, MATLAB, and VisualBasic). There is also an interface with Excel.

Integration

Find
$$\int \cos\left(\frac{x}{a}\right) dx$$.

int(cos(x/a), x);

Answer:
$$a \sin\left(\frac{x}{a}\right)$$

Solution of linear differential equations

Compute an exact solution to the linear ordinary differential equation $\frac{d^2y}{dx^2}(x) - 3y(x) = x$ subject to initial conditions $y(0) = 0, \quad \left.\frac{dy}{dx}\right|_{x=0} = 2$

dsolve({diff(y(x),x,x) - 3*y(x) = x, y(0)=0, D(y)(0)=2}, y(x));

Answer: $y(x) = \frac{7}{18}\sqrt{3}e^{\sqrt{3}x} - \frac{7}{18}\sqrt{3}e^{-\sqrt{3}x} - \frac{1}{3}x$

Root finding

Numerically calculate the root of the equation $e^x = x^2 + 2$ starting at the point $x = -1$; evaluate the answer to 75 decimal digits.

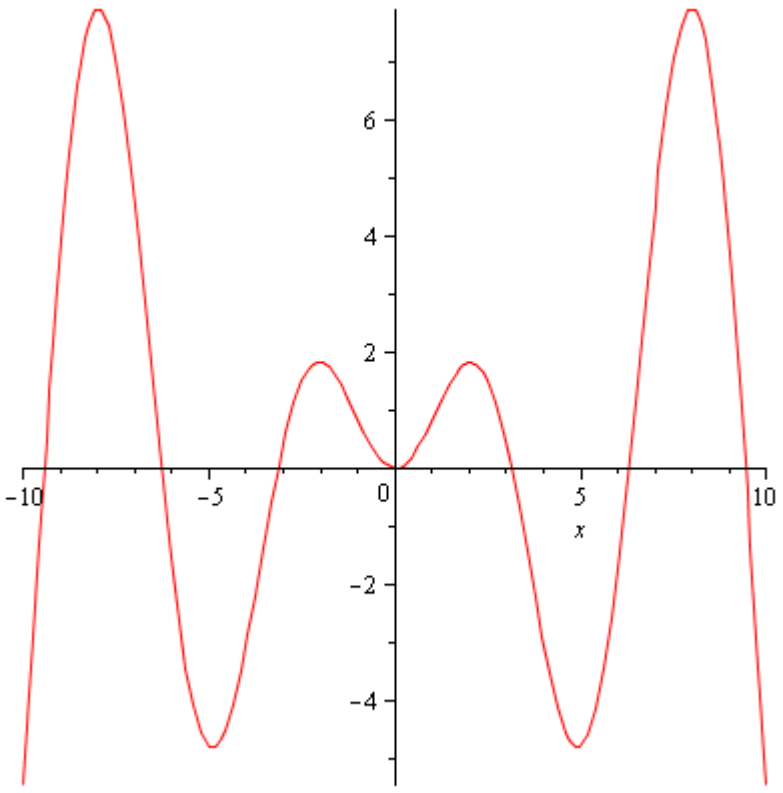
evalf[75](RootOf(exp(x)=x^2+2,x,-1));

Answer: 1.319073676857365354417899109520848464421966780825497669256089004905127076

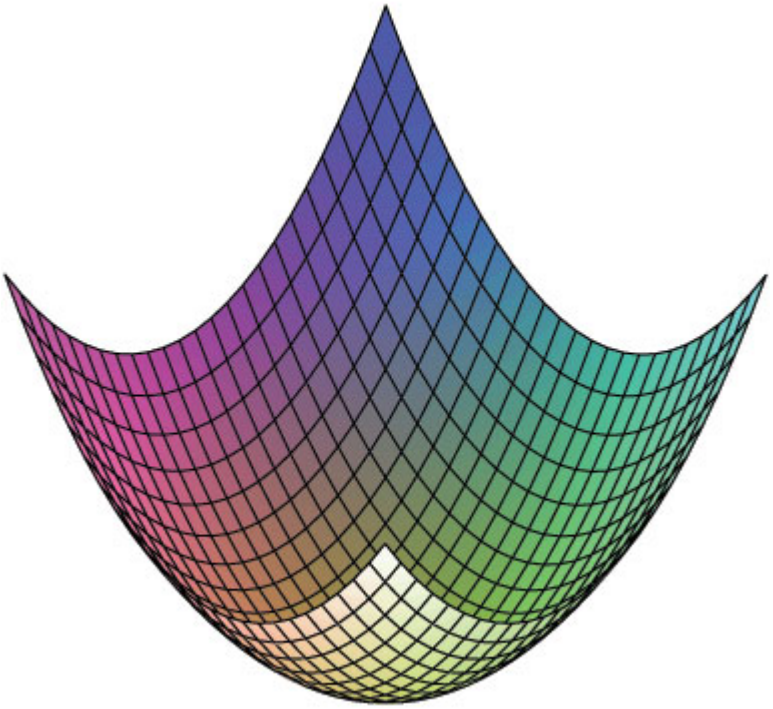
Plotting of function of single variable

Plot $x \cdot \sin(x)$ with x ranging from -10 to 10

plot(x*sin(x),x=-10..10);



Plotting of function of two variables
Plot x2 + y2 with x and y ranging from -1 to 1
plot3d(x^2+y^2,x=-1..1,y=-1..1);



System of partial differential equations

Solve the system of partial differential equations

$$\frac{\partial}{\partial x}v\left(x,t\right)=-u\left(x,t\right)v\left(x,t\right)$$

$$\frac{\partial}{\partial t}v\left(x,t\right)=-v\left(x,t\right)\frac{\partial}{\partial x}u\left(x,t\right)+v\left(x,t\right)\left(u\left(x,t\right)\right)^{2}$$

$$\frac{\partial}{\partial t}u\left(x,t\right)+2\,u\left(x,t\right)\frac{\partial}{\partial x}u\left(x,t\right)-\frac{\partial^{2}}{\partial x^{2}}u\left(x,t\right)=0$$

with $v(x,t)\neq 0$.

```
eqn1:= diff(v(x, t), x) = -u(x,t)*v(x,t):
eqn2:= diff(v(x, t), t) = -v(x,t)*(diff(u(x,t), x))+v(x,t)*u(x,t)^2:
eqn3:= diff(u(x,t), t)+2*u(x,t)*(diff(u(x,t), x))-(diff(diff(u(x,t), x), x)) = 0:
pdsolve({eqn1,eqn2,eqn3,v(x,t)<>0},[u,v]): op(%);
```

Answer:

$$v\left(x,t\right)=e^{\sqrt{\_c_1}x}\_C3\,e^{-\_c_1 t}\_C1+\frac{\_C3\,e^{-\_c_1 t}\_C2}{e^{\sqrt{\_c_1}x}},\quad u\left(x,t\right)=-\frac{\sqrt{\_c_1}\left(\_C1\,\left(e^{\sqrt{\_c_1}x}\right)^{2}-\_C2\right)}{\_C1\,\left(e^{\sqrt{\_c_1}x}\right)^{2}+\_C2}$$

3.2 MATHEMATICA SOFTWARE

Features of Mathematica include

- Elementary mathematical function library
- Special mathematical function library
- Matrix and data manipulation tools including support for sparse arrays
- Support for complex numbers, arbitrary prescription, interval arithmetic and symbolic computation
- 2D and 3D data and function visualization and animation tools
- Solvers for systems of equations, Diophantine equations, ODEs, PDEs, DDEs and recurrence relations
- Numeric and symbolic tools for discrete and continuous calculus
- Multivariate statistics libraries including fitting, hypothesis testing, and probability and expectation calculations on over 100 distributions.
- Constrained and unconstrained local and global optimization
- Programming language supporting procedural, functional, and object oriented constructs
- Toolkit for adding user interface to calculations and applications
- Used for (i) image processing (ii) image recognition
- (iii)visualizing and analysing graphs
- Number theory function library
- Tools for financial mathematics like calculations of bonds, annuities, derivatives, options etc.
- Group theory functions
- Libraries for wavelet analysis on sounds, images and data
- Continuous and discrete integral transforms to mention a few.

Mathematica is split into two parts, the kernel which interprets expressions (Mathematica code) and returns result expressions and front end which allows the creation and editing of Notebook documents containing program code with, formatted text together with results including graphics, tables, and sounds. It also includes development tools such as a debugger, input completion and automatic syntax coloring.

High-performance computing

In recent years, the capabilities for high performance computing have been extended with the introduction of packed arrays sparse matrices to evaluate high-precision arithmetic.

Communication with other applications occurs through a protocol called MathLink. It allows communication between the Mathematica kernel and front-end, and also provides a general interface between the kernel and other applications.

Although Mathematica has a large array of functionality, a number of interfaces to other software have been developed, for use where other programs have functionality that Mathematica does not provide, to enhance those applications, or to access legacy code.

3.3 MATLAB (matrix laboratory)

Matlab is a numerical computing environment and fourth generation programming language. Developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java and Fortran.

MATLAB was created in the late 1970s by Cleve Moler, the chairman of the computer science department at the University of New Mexico. He designed it to give his students access to LINPACK and EISPACK without having to learn Fortran. It soon spread to other universities and found a strong audience within the applied mathematics community.

MATLAB was first adopted by control design engineers, but quickly spread to many other domains. It is now also used in education, in particular the teaching of linear algebra and numerical analysis, and is popular amongst scientists involved with image processing.[3]

Variables

Variables are defined with the assignment operator, =. MATLAB is a programming language. It is a weakly typed language because types are implicitly converted. It is a dynamically typed language because variables can be assigned without declaring their type, except if they are to be treated as symbolic objects, and that their type can change. Values can come from constants, from computation involving values of other variables, or from the output of a function. For example:

```
>> x = 17
x =
17
```

```
>> x = 'hat'
x =
hat
>> y = x + 0
y =
104 97 116
>> x = [3*4, pi/2]
x =
12.0000 1.5708
>> y = 3*sin(x)
y =
-1.6097 3.0000
```

Vectors/matrices

MATLAB is a "Matrix Laboratory", and as such it provides many convenient ways for creating vectors, matrices, and multi-dimensional arrays. In the MATLAB vernacular, a vector refers to a one dimensional (1×N or N×1) matrix, commonly referred to as an array in other programming languages. A matrix generally refers to a 2-dimensional array, i.e. an m×n array where m and n are greater than or equal to 1. Arrays with more than two dimensions are referred to as multidimensional arrays. Arrays are among the fundamental types and several standard functions natively support array operations which are optimized to work without explicit loops. Therefore the MATLAB language is also an example of array programming language.

MATLAB provides a simple way to define simple arrays using the syntax: init:increment:terminator. For instance:

```
>> array = 1:2:9
array =
1 3 5 7 9
```

defines a variable named array (or assigns a new value to an existing variable with the name array) which is an array consisting of the values 1, 3, 5, 7, and 9. That is, the array starts at 1 (the init value), increments with each step from the previous value by 2 (the increment value), and stops once it reaches (or to avoid exceeding) 9 (the terminator value).

```
>> array = 1:3:9
array =
1 4 7
```

the increment value can actually be left out of this syntax (along with one of the colons), to use a default value of 1.

```
>> ari = 1:5
ari =
1 2 3 4 5
```

assigns to the variable named ari an array with the values 1, 2, 3, 4, and 5, since the default value of 1 is used as the incrementer.

Indexing is one-based, which is the usual convention for matrices in mathematics, although not for some programming languages.

Matrices can be defined by separating the elements of a row with blank space or comma and using a semicolon to terminate each row. The list of elements should be surrounded by square brackets: []. Parentheses: () are used to access elements and subarrays (they are also used to denote a function argument list).

```
>> A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
A =
16 3 2 13
5 10 11 8
9 6 7 12
4 15 14 1
>> A(2,3)
ans =
11
```

Sets of indices can be specified by expressions such as "2:4", which evaluates to [2, 3, 4]. For example, a submatrix taken from rows 2 through 4 and columns 3 through 4 can be written as:

```
>> A(2:4,3:4)
ans =
11 8
7 12
14 1
```

A square identity matrix of size n can be generated using the function eye, and matrices of any size with zeros or ones can be generated with the functions zeros and ones, respectively.

```
>> eye(3)
ans =
1 0 0
0 1 0
```

0 0 1

>> zeros(2,3)

ans =

0 0 0

0 0 0

>> ones(2,3)

ans =

1 1 1

1 1 1

Most MATLAB functions can accept matrices and will apply themselves to each element. For example, mod(2*J,n) will multiply every element in "J" by 2, and then reduce each element modulo "n". MATLAB does include standard "for" and "while" loops, but using MATLAB's vectorized notation often produces code that is easier to read and faster to execute. This code, excerpted from the function magic.m, creates a magic square M for odd values of n (MATLAB function meshgrid is used here to generate square matrices I and J containing 1:n).

```
[J,I] = meshgrid(1:n);
A = mod(I+J-(n+3)/2,n);
B = mod(I+2*J-2,n);
M = n*A + B + 1;
```

Semicolons

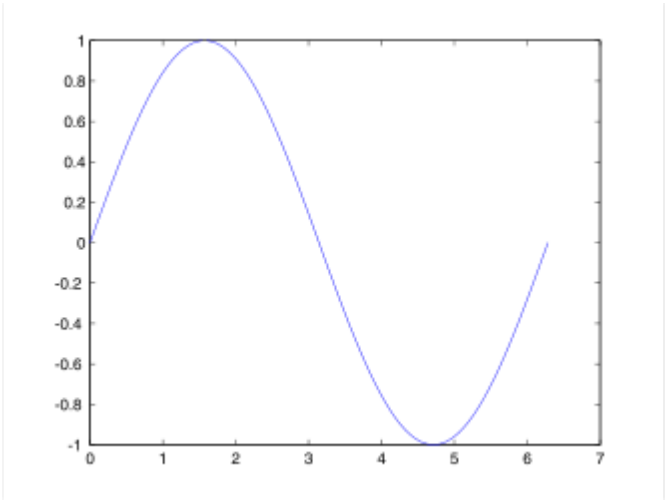
Unlike many other languages, where the semicolon is used to terminate commands, in MATLAB the semicolon serves to suppress the output of the line that it concludes (it serves a similar purpose in Mathematica)

Graphics

Function plot can be used to produce a graph from two vectors x and y. The code:

```
x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y)
```

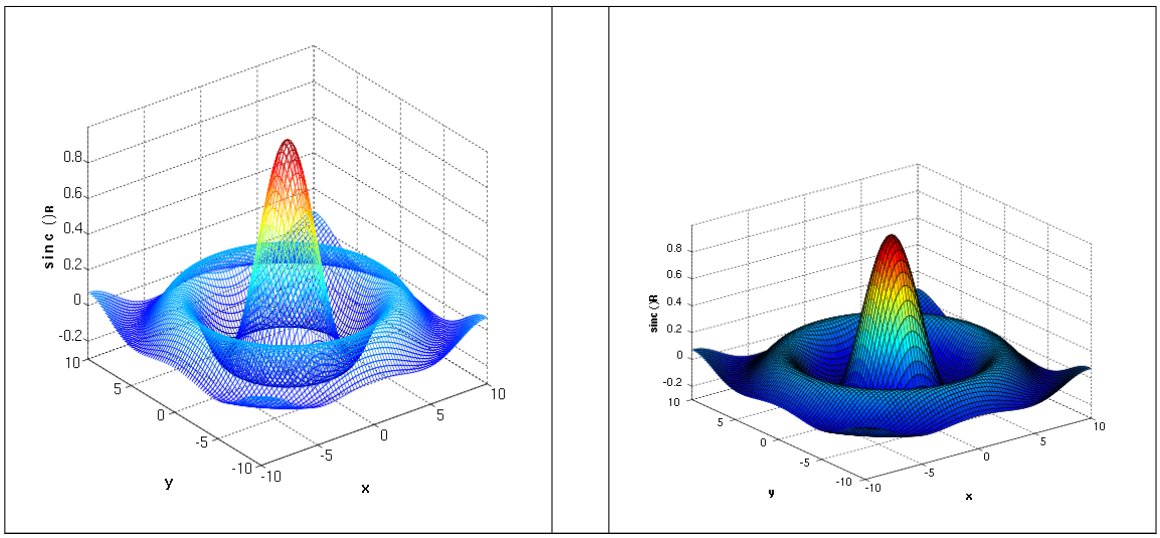
produces the following figure of the sine function:



Three-dimensional graphics can be produced using the functions surf, plot3d or mesh.

[X,Y] = meshgrid(-10:0.25:10,-10:0.25:10);
f = sinc(sqrt((X/pi).^2+(Y/pi).^2));
mesh(X,Y,f);
axis([-10 10 -10 10 -0.3 1])
xlabel('{\bfx}')
ylabel('{\bfy}')
zlabel('{\bfsinc} ({\bfR})')
hidden off
		[X,Y] = meshgrid(-10:0.25: 10,-10:0.25:10);
		f = sinc(sqrt((X/pi).^2+(Y/pi).^2));
		surf(X,Y,f);
		axis([-10 10 -10 10 -0.3 1])
		xlabel('{\bfx}')
		ylabel('{\bfy}')
		zlabel('{\bfsinc} ({\bfR})')

This code produces a wireframe 3D plot of the two-dimensional unnormalized sine function:		This code produces a surface 3D plot of the two-dimensional unnormalized sine function:



Structures

MATLAB supports structure data types. Since all variables in MATLAB are arrays, a more adequate name is "structure array", where each element of the array has the same field names. In addition, MATLAB supports dynamic field names (field look-ups by name, field manipulations etc). Unfortunately, MATLAB JIT does not support MATLAB structures, therefore just a simple bundling of various variables into a structure will come at a cost.

Function handles

MATLAB supports elements of lambda-calculus by introducing function handles, or function references, which are implemented either in .m files or anonymous/nested functions.

Secondary programming

MATLAB also carries secondary programming which incorporates the MATLAB standard code into a more user friendly way to represent a function or system.

Classes

MATLAB supports classes, however the syntax and calling conventions are significantly different than in other languages. MATLAB supports value classes and reference classes, depending if the class has handle as super-class (for reference classes) or not (for value classes).

Depending if a class is declared as value or reference, method call behavior is different. For example, a call to a method

object.method();

can alter any variables of object variable only for reference classes.

Object-oriented programming

MATLAB's support for object-oriented programming includes classes, inheritance, virtual dispatch, packages, pass-by-value semantics, and pass-by-reference semantics.

classdef hello

methods

function doit(this)

disp('Hello!')

end

end

end

When put into a file named hello.m, this can be executed with the following commands:

>> x = hello;

>> x.doit;

Hello!

Interfacing with other languages

MATLAB can call functions and subroutines written in the C programming language or Fortran. A wrapper function is created allowing MATLAB data types to be passed and returned.

Libraries written in Java, ActiveX or NET can be directly called from MATLAB and many MATLAB libraries are implemented as wrappers around Java or ActiveX libraries.

3.4 REPSN SOFTWARE:

Repsn was been developed by Vahid Dabbaghian at the Department of Mathematics, Simon Fraser University, Burnaby, British Columbia V5A 1S6 Canada.

Repsn is a package for computing matrix representations in characteristic zero of finite groups. Most of the functions in Repsn have been written according to the algorithm described in Dabbaghian-Abdoly [1] and Dabbaghian-Abdoly and Dixon [2] (see [3].

For constructing representations of simple groups and their covers we use the algorithm described in Dixon [4]. To use this algorithm for constructing a representation of a group G affording an irreducible character chi of G, we need to have a subgroup H of G such that the restriction of chi to H has a linear constituent with multiplicity one. In this case we say H is a character subgroup relative to chi (or a chi-subgroup). A chi-subgroup for each irreducible character chi of degree less than 100 of simple groups and their covers are listed in [3] and [5].

All Repsn functions are written entirely in the GAP language. It is proved in [3] and [2] that the algorithm is correct for any group with a character of degree less than 100. Indeed, if the group is solvable, there is no restriction on the character degree. In practice the program is quite fast when the degree is small, but can be very slow when it is necessary to call one of the subprograms which extend irreducible representations. In the latter case the number of element wise operations required to extend a representation of degree d is proportional to d6.

Repsn is implemented in the GAP language, and runs on any system supporting GAP4. The Repsn package is loaded into the current GAP session with the command

gap> LoadPackage("repsn");

One could install the Repsn package on GAP4.3. In this case it is loaded with the command

gap> RequirePackage("repsn");

A package for solving polynomial equations is contained in [12]. This software is very useful. We recall that one of the most classical problems of mathematics is to solve systems of polynomial equations in several unknowns. Today, polynomial models are ubiquitous and widely applied across the sciences. They arise in robotics, coding theory, optimization, mathematical biology, computer vision, game theory, statistics, machine learning, control theory, and numerous other areas.

3.5 SINGULAR (typeset Singular)

Singular is a computer algebra system for polynomial computations with special emphasis on the needs of commutative algebra, and singularity theory (see Golubitsky et al 13]). SINGULAR is a free software released under the GNU General Public License. Problems in non-commutative algebra can be tackled with the SINGULAR offspring PLURAL. SINGULAR was designed by Gert-Martin Greuel, Gerhard Pfister and Hans Schönemann [12]

3.6 Conclusion

I have highlighted in this paper only some of the available software in Physical Sciences. You must have observed that in recent times every software are trying to fit into MATLAB. Mathematica is trying to make itself a toolbox for MATLAB. As noted by Griffiths [10] here is a glimpse of the power and flexibility of the Matlab system:

- Matlab is an interactive systems for doing numerical computations
- Matlab relieves you of a lot of the mundane tasks associated with solving problems. This allows you to spend more time thinking and encourages you to experiment.
- Matlab makes use of highly respected algorithms and hence you can be confident about your results
- Powerful operations can be performed using just one or two commands
- You can build your own set of functions for a particular application
- Excellent graphics facilities are available and the pictures can be inserted into LaTeX and Word documents.

REFERENCES

1. Vahid Dabbaghian (2008), Repsn a GAP4 package for constructing representation of finite groups version 3.0.1, http://mats.sfu.ca/ vdabbagh
2. Ramsaroop A. and Kanny K. (2010), Using MATLAB to Design and Analyse Composite Laminates, Engineering 2010, 2, 904-916
3. Oostenveld R., Fries P, Maris e. and Mathljs J. (2011), FieldTrip: Open source software for Advanced Analysis of MEG, EEG and invasive electrophysiological data, Computational Intelligence and Neuroscience Vol 2011 (2011) doi:10.1155/2011/156869
4. Vahid Dabbaghian-Abdoly. An algorithm to construct representations of finite groups. Ph.D.thesis, Dept. Mathematics, Univ. Carleton, 2003
5. Vahid Dabbaghian-Abdoly (2005). An algorithm for constructing representations of finite groups. J. Symbolic Comput., 39:671– 688
6. Vahid Dabbaghian-Abdoly (2006). Constructing representations of finite simple groups and central covers. Canad. J. Math., 58:23 – 38
7. Vahid Dabbaghian-Abdoly (2007). Constructing representations of higher degrees of finite simple groups and covers. Math. Comp., 76:1661 – 1668
8. Vahid Dabbaghian and John D. Dixon(2008). Computing matrix representations. Math. Comp., 9 pages

9. John D. Dixon (1993). Constructing representations of finite groups. In Groups and Computation,volume 11 of Dimacs Series in Discrete Mathematics and Theoretical Computer Science, pages 105–112

10. Griff iths , D. F. , An Introduction to MATLAB, University of Dundee (2005)

11. Wikipedia, the free encyclopedia.

12. Sturmfels, B. Solving systems of polynomial equations, University of California, Berkely (2002)

13. Golubitsky, M., Schaeffer, D. C. Singularities and Groups in Bifurcation Theory, Vol. 1 Appl. Math. Sciences 51, Springer-Verlag, New York (1985)

MATLAB/SCILAB Training Course, Vol. 1, pp. 34–65, 2011

Received 31 October 2011

CHAPTER 4

DESIGN OF EXPERIMENTS FOR EFFICIENT DATA ANALYSIS/VISUALIZATION

by

CHUKWU W. I. E
Associate Professor in Statistics/Operations Research
University of Nigeria, Nsukka

4.1 INTRODUCTION

4.1.1 Introduction

Experiments are performed by investigation in virtually all fields of inquiry, usually to discover something about a particular process or system. Literally, an experiment is a test. More formally, we can define an experiment as a set of test or series of tests in which purposeful changes are made to the input variables of a process or system so that we may observe and identify the reasons for changes that may be observed in the output response (Douglas 2001)

This paper is about planning and conducting purposeful experiments and analyzing the resulting data in other that valid and objective conclusions are obtained. Our focus here is on experiments in Agriculture, Engineering, Physical and biological sciences. In Engineering, experimentation plays an important role in new product design, manufacturing process development and process improvement. In Agriculture, experimentation plays an important role in obtaining new methods of preserving agricultural products. In the western world today, various experiments are now carried out to obtain new methods of processing eggs so as to reduce the number of germs on the eggs in a pack of thirty

and also some experiments are going on, on extension to a reasonable point of the shell life of eggs. Furthermore in the field of agriculture, new verities of resistant crops are also produced through experimentation. Sometimes in the field of engineering, the entire objective could, in many cases be to develop a robust process, that is, a process that is affected minimally by external sources of variability.

Design of experiment, like any other scientific discipline, has its peculiar terminology, methodology and research procedures. The title of this scientific discipline itself clearly indicates that it deals with experimental methods. A large number of experiments are done in the area of research, development and optimization of a system. These researches are done in laboratories, pilot plants, agricultural lots, clinics etc. An experiment may be physical, psychological or model based. It may be performed directly on the subject or on its model. When the model describes the subject precisely enough, the experiment on the subject is generally replaced by an experiment on the model. Lately, due to a rapid development of computer technology, physical models are more frequently replaced by abstract mathematical ones.

An experiment takes a central place in science, engineering, agriculture and medicine, particularly nowadays, due to the complexity of problems these disciplines deal with. The question of efficiency of using an experiment is therefore of paramount importance.

Furthermore there are three key words in the title of this paper: experiment, design and visualization. We shall now explain the meaning of the three key words as they appear in the title of this paper. Experiment can be understood to be an act of conducting a controlled test or investigation. The word controlled means that most, if not all, of the conditions that happened or were used in the experiment are known or regulated. In all the disciplines mentioned earlier on, an experimental research is conducted to answer a particular question or solve a particular problem. In experimental research, different kinds or levels of a particular factor or several factors are evaluated.

The second word is design which may mean arrangement. In every field of research process, proper design is important because we want to establish or find the true results without any doubt in mind. With improper or wrong design, results may not be convincing or reliable

The third is visualization which (McGee 1979 P. 893) defines as 'the comprehension of the arrangement of elements within a visual stimulus pattern and attitude and aptitude to remain unconfused by the changing orientation in which a configuration may be represented. For example, when a swimmer is diving, even though he or she may be turning and twisting he or she knows where the water is, a stunt pilot knows where the ground is during his or her maneuvers.

Research means systematic investigation to establish new facts or principles. It is systematic because all the activities are planned and executed based on rules so everything can be repeated. The term, establish new facts or principles indicates that research is done to prove or disprove something that has been done before. The procedure for research is generally known as the scientific method which, although difficult to define precisely, usually involves the following elements.

- Observed facts. Science and engineering practices is said to begin with observation, through which many facts are established

- Hypothesis. A consideration of the body of facts about a subject leads to the establishment of an hypothesis- a tentative idea as to how the facts are to be interpreted and explained
- Experiment: The experiment is a trial designed to test the validity of the proposed hypothesis.
- Results and their interpretation: The results of the experiment establish additional facts. The interpretation of these facts in the light of what is known already leads to the support, rejection or alteration of the hypothesis thus the same series of steps again.

4.1.2 CHARACTERISTICS OF A WELL PLANNED EXPERIMENT

- Simplicity : The selection of treatments and the experimental arrangement should be as simple as possible, consistent with the objectives of the experiment
- Degree of precision: The probability would be high that the experimenter will be able to measure treatment differences with the degree of precision the experimenter desires. This implies an appropriate design and sufficient replication
- Absence of systematic error: The experiment must be planned to ensure that experimental units receiving one treatment, in no systematic way differ from those receiving another treatment so that an unbiased estimate of each treatment can be obtained.
- : Range of validity of conclusion: Conclusions should have as wide a range of validity as possible. An experiment replicated in time and space would increase the range of validity of the conclusions that could be drawn from it. A factorial set of treatment is another way for increasing the range of validity of an experiment. In a factorial experiment the effects of one factor are evaluated under varying levels of a second factor.
- Calculation of degree of uncertainty: In every experiment there is always some degree of uncertainty of conclusions. The experiment should be designed so that it is possible to calculate the probability of obtaining the observed results by chance alone

4.1.3 1.3 PROCEDURE FOR EXPERIMENTATION

In planning and conduct of an experiment there are a number of considerations that should be carefully thought through if the experiment is to be a success. Some of the more important steps to be taken are:

1. Definition of the problem. The first step in problem –solving is to state the problem you are dealing with clearly and concisely. Once the problem is understood you should be able to formulate questions which, when answered will lead to solution.

2. Statement of the objectives. This may be in the form of question to be answered, the hypothesis to be tested, or the effects to be estimated. Objectives should be written out in precise terms. When this is done the experimenter is able to plan his experimental procedures more effectively. When there is more than one objective, prioritize them as this might have a bearing on the experimental design. In stating objective don't be vague or ambition.

3. Selection of treatments. The success of the experiment rests on the careful selection of treatments, whose evaluation will answer the questions posed.

4. Selection of experimental material: In selecting experimental material, consider the objective of the experiment and the population about which inferences are to be made The material used should be representative of the population on which you wish to test y our treatments.

5. Selection of experimental design. Here again a consideration of objectives is important, but a general rule would be to choose the simplest design that is likely to provide the precision you require.

6. Selection of units of observation and the number of replication. For field experiment in agriculture with plants, this means deciding on the size and shape of the plots. For experiments with animals, this means deciding on the number of animals to consider as a unit. In engineering processing plants a unit could be 2000 bolts etc. Here plot size and the number of replication should be chosen to produce the required precision of treatment estimates

7. Consideration of data to be collected. The data collected should properly evaluate treatment effects in line with the objectives of the experiment. In addition, consideration should be given to collection of data that will explain why the treatments perform as they do.

8. Outlining statistical analysis and summarization of results. Write out the sources of variation and associated degrees of freedom in the analysis of variance. Include the various F tests you may have planned. Consider how the results might be used and prepare possible summary tables or graphs that will show the effects you expect. Compare these expected results to the objectives of your experiment to see if the experiment will give the answers you are looking for.

9. Conducting the experiment. In the conduct of the experiment, use procedures that are free from personal biases or favouritisms. Make use of the experimental design in collecting data, so that differences among individuals or differences associated with order of collection can be removed from experimental errors. Organize the collection of your data to facilitate analysis and to avoid errors in recopying. If it is necessary to copy data, check the copied figures against the originals immediately

10. Analyzing data and interpreting result. All data should be analyzed as planned and the results interpreted in the light of the experimental conditions, hypothesis tested and the relation of the

results to facts previously established. Understand that there is always a possibility of making a wrong conclusion and so put into consideration the consequences of making an incorrect decision.

4.1.4 SOME DEFINITIONS IN THE EXPERIMENT AND SURVEY DATA

Factor: A factor of an experiment is a controlled independent variable whose levels are set by the experimenter. The effect of starch on paper strength experiment-Starch is a factor.

Treatment: A treatment is a level (amount) of factor applied to the experimental units. For example, an experiment is divided into four units; each part is 'treated' with 10mg, 20mg, 30mg and 40mg of the same starch N. Each amount of N is a treatment.

Variable: All the factors, their levels and all the measured traits (responses) are variables

Independent or class variables: Factors applied to the experimental units are independent variables

Response or dependent variables: Measured or observed traits in the experiment. The strength of the paper measured after 'treated' with starch is a response (or dependent) variable.

Main effects: This is the simple effect of a factor on a dependent variable. It is the effect of the factor alone averaged across the levels of other factors. Example: In the starch experiment, it was found that on average, the starch increased the strength (main effect) compared to the control (no starch).

Interaction effect: An interaction is the variation among the differences between means for different levels of one factor over different levels of another factor. Example: Lets say starch and latex both significantly increased the paper strength (main effects).

However, when starch and latex applied to the same experimental unit, it was found that the increase in paper strength was even higher. The paper got the benefits of both factors, plus a bonus.

Experimental (or Sampling) Unit: A unit is a person, animal, plant or thing, which is studied by a researcher; the basic objects upon which the study or experiment is carried out. For example, a person; a sample of soil; a pot of seedlings are units.

Population: A population is any entire collection of people, animals, plants from which we may collect data. It is the entire group we are interested in, which we wish to describe or draw conclusions about.

Sample: A sample is a group of units selected from a larger group (the population). By studying the sample it is hoped to draw valid conclusions about the larger group. A sample is generally selected for study because the population is too large to study in its entirety.

Randomization: The sample should be representative of the general population. This is often best achieved by random sampling. For each population there are many possible samples.

Parameter: A parameter is a value, usually unknown (and which therefore has to be estimated). It is used to represent a certain population characteristic. For example, the population mean is a parameter that is often used to indicate the average value of a quantity. Parameters are often assigned Greek letters (e.g. sigma).

Statistic: A statistic is a quantity that is calculated from a sample of data. For example, the mean, a variance and a standard deviation calculated from a sample are statistics. Statistics are assigned Roman letters (e.g. s for standard deviation). The value of a statistic will vary from sample to sample.

Sampling Distribution: The sampling distribution is the probability distribution or probability density function of the statistic. Derivation of the sampling distribution is the first step in calculating a confidence interval or carrying out a hypothesis test for a parameter.

Example, suppose that x1, x2, ..., xn are a simple random sample from a normally distributed population with expected value μ and known variance σ^2 . Then the sample mean x is normally distributed with expected value μ and variance σ^2 /n

Estimate: An estimate is an indication of the value of an unknown quantity based on observed data. Example, suppose we want to know the average height of CNR students (population). We can use an estimate of this population mean by calculating the mean of a sample of students. Estimators of population parameters are sometime distinguished from the true value by using the symbol 'hat'. For example, σ = true population standard deviation. σ = estimated (from a sample) population standard deviation.

4.1.5 TOOLS FOR DEVELOPING OF EXPERIMENTAL DESIGN

Randomization

Treatments should be allocated to the experimental units randomly. In the additive experiment, we would like to treat 10 samples per additive. The additives should be assigned 1 to 10 paper samples randomly. Why is it important?

1. First, randomization makes sure that the sample is a representative of the population (whole paper produced in the mill).
2. Secondly, randomization eliminates the effect of systematic biases. Le us say the humidity in the mill increases during the day. If we do not randomly assign additives to the sample papers during the day, the humidity may have a confounding effect on the paper strength.

Replication

This is the number of experimental units (paper samples) measured for each treatment. Increasing the number of replications means collecting more information about the treatments. In the additives example, we tested 10 paper samples for each additive. Thus the number of replications per treatment is 10. Replication provides the variability in the response variable that is not associated with the treatment effects. Increasing the number of replications increases the reliability of the outcome.

Blocking

Blocking is a technique used to eliminate the effect of a confounding factor. Blocks are group of experimental units sharing a common level of a confounding variable. For example, in the additives experiment, we assume that humidity is an external factor and affecting the paper strength. If we treat all 10 samples with starch in the morning, and treat another 10 random samples with latex in the afternoon, we cannot eliminate the affect of humidity. The effect of treatments will be confounded. There are different blocking techniques. The most commonly blocking is the Randomized Complete Blocks. It is random, because the experimental units are randomly assigned to each block. It is complete, because each treatment is included in every block. Assignment of experimental units to the blocks and to the treatments must be random.

4.2 DIFFERENT TYPES OF DESIGNS APPLIED IN EXPERIMENTAL LAYOUTS

In this section we shall briefly consider the following types of designs

1. Completely randomized design
2. The completely randomized block design
3. Incomplete randomized design
4. The Latin Square design
5. The factorial design
6. The nested design

4.2.1 COMPLETELY RANDOMIZED BLOCK DESIGN

In empirical or experimental research it is necessary to determine system stability and reproducibility of the results. In the case of an experiment with a larger number of design points there appears to be a problem of providing the same conditions for doing the design points. The experiment is therefore designed so as to do research in groups of design point's blocks, where equality of conditions is higher than in the complete field of research. In this way we can single out the effects of inequality of conditions from other factors. The design of experiments that provide this are called completely randomized block design. This design is the simplest design and it is set up by assigning treatments at random to a previously determined set of experimental units. It may not be the most efficient design for some type of engineering problem but it may be the most workable arrangement for testing certain other types of treatments on some other scientific problems. Usually they are used in research with a single factor, any number of treatments may be tested in this design if they belong to the single-factor design group. Results of an experiment done by a randomized complete block design are analyzed by the method of analysis of variance. Since these randomized blocks are applied to single out inequality effects of a research subject from factor effects, the variance of analysis confidence is increased as experimental error is diminished. The block denotes the part of design points where experimental error is lower than in the experiment as a whole. To screen out the effects of systematic errors, the effects of factor-level variations are researched in each block by random order. This is the origin of the term randomized complete-block design. These blocks originate from studies in agronomy, for in it there appeared the most drastic case of inequality of agricultural lots where experiments have been done. To eliminate this inequality, the lots have been divided into blocks or more equal areas. The size and number of blocks primarily depends on the research subject and possibility to equalize experimental conditions, and they are two counter-balanced requirements on which processing confidence of results depends. A small number of blocks means simple calculations of analysis of variance but also greater lack of confidence due to their inequality. When designs of completely randomized blocks are used, an assumption is introduced that outcome levels may be different in different blocks but that relative factor effects are the same in all blocks. This assumption means in practice that there is no interaction between blocks and factors, i.e. even if it exists it is negligible in relation to the factor effect. Interaction is part of experimental error and when it is large, inferences of analysis of variance are not certain. One of the basic objectives of designing experiments by completely randomized blocks, when an experiment is done in full-scale plants, is that the time element is reduced. In a normal operation of a chemical plant, in batches or continuously, systematic variations in product properties appear. Sometimes these variations may be explained by seasonal influences, such as change in temperature of cooling water, for example, a change in quality of raw materials, etc. However, frequently there are neither explanations for the mentioned fluctuations nor can they can be controlled. The question is not about temporary, random variations that are considered normal in production, but about slow changes in the average level. Therefore, in designing an experiment in such plants, one must take care of separating the effects of temporary trends. This does not mean that the designs of completely randomized blocks are limited only to full-scale. But on the contrary, they are limited to labs and

pilot plants where trials for one experiment are done in a longer time period and where there exists a probability of systematic variations and trends

Example 1

Four different feed rates were investigated in an experiment on a CNC machine producing a component part used in an aircraft auxiliary power unit. The manufacturing engineer in charge of the experiment knows that a critical part dimension of interest may be affected by the feed rate. However, prior experience has indicated that only dispersion effects are likely to be present. That is, changing the feed rate does not affect the average dimension, but it could affect dimensional variability. The engineer makes five production runs at each feed rate and obtains the standard deviation of the critical dimension (in 10-3 mm) The data are shown below. Assume all the runs are made in random order.

Table 4.1: ANALYSIS OF RESULTS OF AN EXPERIMENT IN WHICH COMPLETELY RANDOMIZED BLOCKS DESIGN IS APPLIED

Feed rate / (in/mm)	Production run				
	1	2	3	4	5
10 12 14 16	.0.09 0.06 0.16 0.19	0.10 0.09 0.08 0.13	0.13 0.12 0.08 0.15	0.08 0.07 0.05 0.20	0.07 0.12 0.06 0.11

Since the experimental results, by design of completely randomized blocks, are processed by analysis of variance, experimental results of randomized blocks will be presented as a two-way classification and notation. Here we state that the measured values or responses are marked by y_{ij} and factors are marked by X_{ij} . The Design of completely randomized structure is given in 4.2.

The experiment is done by the design shown in Table 4.2, and the obtained results processed by analysis of variance. Definitions with calculation forms of analysis of variance are shown in Table 4.3. Although block effects are less interesting for the research objective, the large variance value between the blocks means that division into blocks was justified, i.e. that residual variance has been reduced, which contributed to a higher precision of the experiment. A high variance value between the blocks means at the same time that one should find out the causes for such an expressed inequality of experiment conditions. It should be noted that analysis of variance in Table 2. comprises performing the experiment by design of randomized blocks without repeating the measurements. The basic assumption of this kind of design of experiments without repeating measurements is that each measured result may be described by this model.

Table 4.2: Structure of design of completely randomized blocks

Factor level	Blocks					Average value
	1	2	3	⋯	J	
1	Y11	Y12	Y13	⋯	Y1J	Y1.
2	Y21	Y22	Y23	⋯	Y2J	Y2.
3	Y31	Y31	Y33	⋯	Y3J	Y3.
.	.	.	.	⋯	.	.
I	YI1	YI2	YI3	⋯	YIJ	YI.
Average number	Y.1	Y.2	Y.3	⋯	Y.J	Y..

$$Y_{ij} = \mu + \alpha_i + \beta_j + \varepsilon_{ij} \qquad i = 1, 2, 3, \ldots l \tag{4.1}$$

where

μ is actual mean effect; β is actual block effect; α is actual factor effect; and ε is experimental error with normal distribution $\mathrm{N}\left(0, \sigma^2\right)$

Experiments may be done by design of completely randomized blocks and by repeating measurements in which case analysis of variance has a different form

It should be noted that the number of measurement replications in the matrix of design of completely randomized blocks is marked by K. A distinction should also be made between mean squares for measurement error + experimental error and measurement error. Often this sum of measurement and experimental errors is just called experimental error, and measurement error sampling error. To check significance of the factor effect, the mean square of joint error or experimental error MSCR is used.

We use the SPSS software to demonstrate the solution of the example posed above for solving results obtained from a complete randomized experiments if the effects of main effects are the items of interest.. The ANOVA table for the analysis is give as

4.2.2 Factorial Completely randomized design

In this section we shall consider another type of design that can be used when joint effects are required. Here we start with factorial experiments conducted in completely randomized designs that are especially useful for evaluating joint factor effects. Factorial experiments include all possible factor–level combinations in the experimental design

Table 4.3: Analysis of variance of completely randomized blocks

Source of variation	df	Sum of squares definition	Sum of squares calculation	Mean squares	Test statistic
Between columns of blocks	J-1	$SS_C = I\sum_j (Y_{\cdot j} - Y_{\cdot\cdot})^2$	$SS_c = \frac{\sum_j Y_{\cdot j}^2}{I} - \frac{Y_{\cdot\cdot}^2}{IJ}$	$MS_C = \frac{SS_C}{J-1}$	$\frac{MS_C}{MS_E}$
Between rows of blocks	I-1	$SS_R = J\sum_i (Y_{i\cdot} - Y_{\cdot\cdot})^2$	$SS_c = \frac{\sum_I Y_{I\cdot}^2}{J} - \frac{Y_{\cdot\cdot}^2}{IJ}$	$MS_R = \frac{SS_C}{I-1}$	$\frac{MS_R}{MS_E}$
Residual variance of errors	(J-1)(I-1)	$SS_E = \sum_i \sum_j (Y_{ij} - Y_{\cdot j} - Y_{i\cdot} + Y_{\cdot\cdot})^2$	$SS_E = SS_T - SS_C - SS_R$	$MS_E = \frac{SS_E}{(I-1)(J-1)}$	-
Total	IJ-1	$SS_T = \sum_i \sum_j (Y_{ij} - Y_{\cdot\cdot})^2$	$SS_T = \sum_j \sum_i Y_{ij}^2 - \frac{Y_{\cdot\cdot}^2}{IJ}$	..	..

This type of randomized designs are appropriate when there are no restrictions on the order of testing, and/or when all the experimental units to be used in the experiment can be regarded as homogeneous. The calculation of factor effects is shown to be a valuable aid in interpreting the influence of factor levels on a response.

In a completely randomized design all the factor–level combinations in the experiment are randomly assigned to experimental units, if appropriate, or to the sequence of test runs. Randomization is important in any experimental design because an experimenter cannot always be certain that every major influence on a response has been included in the experiment. Even if one can identify and control all the major influences on a response, unplanned complications are common. Instrument drift, power surges, equipment malfunctions, technician or operator errors, or a myriad of other undetectable influences can bias the results of an experiment. Randomization does not prevent any of the above experimental complications from occurring. Randomization affords protection from bias by tending to average the bias effects over all levels of the factors in the experiment. When comparisons are made among levels of a factor, the bias effects will tend to cancel and the true factor effects will remain. Randomization is not a guarantee of bias-free comparisons, but it is certainly inexpensive insurance. There are numerous ways to achieve randomization in an experimental design. Any acceptable randomization procedure must, however, adhere to procedures that satisfy the definition given below

Randomization. Randomization is a procedure whereby factor–level combinations are (a) assigned to experimental units or (b) assigned to a test sequence in such a way that every factor–level combination has an equal chance of being assigned to any experimental unit or position in the test sequence.

. With this definition of randomization, the steps used to construct a completely randomized design

are given in below. Note that the randomization procedure described below is equivalent to obtaining a simple random sample without replacement

Steps used to construct a completely randomized design

1. Enumerate all factor–level combinations. Include repeat tests.
2. Number the factor–level combinations, including any repeat tests, sequentially from 1 to N.
3. From a random-number table or from a computer-generated sequence of random numbers, obtain a random sequence of the integers 1 to N.
4. Assign the factor–level combinations to the experimental units or conduct the test runs in the order specified by the random number sequence. If both a randomized assignment to experimental units and a randomized test sequence are to be included in the design, use two random number sequences.

Benefits of using factorial design

Several benefits accompany the factorial experiments.. First, the inclusion of each factor level with a variety of levels of other factors means that the effects of each factor on the response are investigated under a variety of different conditions. This allows more general conclusions to be drawn about the factor effects than if each factor effect were studied for a fixed set of levels of the other factors.

A second benefit of these designs is that the randomization protects against unknown biases, including any unanticipated or unobservable "break-in" effects due to greater or lesser care in conducting the experiment as it progresses. Note too that the randomization of the repeat tests ensures that responses from repeat tests give a valid estimate of the experimental error of the test runs. If "back-to-back" repeat tests are conducted, the estimate of experimental error can be too small because any variability associated with setting up and tearing down the equipment would not be present.

A third major benefit of factorial experiments conducted in completely randomized designs is the ability to investigate joint factor effects. There are joint factor effects among the factors in the torque study. The importance of planning experiments so that joint factor effects can be measured is the topic of discussion in the next section.

Statistical Analysis of the Fixed Effect Factors

In fixed effect model where two factors are of interest , the a level "a" level of the factor A and the "b" levels of the factor B are specifically chosen by the experimenter and inferences are confined to these levels only. In this model it is usual to define the effects α_i, β_j and $(\alpha\beta)_{ij}$ as deviations from the mean, so that

$$\sum_{i=1}^{n} a_i = 0, \sum_{j=1}^{n} \beta_j = 0,$$
$$\sum_{i=1}^{n} (\alpha\beta)_{ij} = 0, \sum_{j=1}^{n} (\alpha\beta)_{ij} = 0$$

The sum of squares identity for a 2 factor ANOVA are given below

$$\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n}\left(y_{ijk} - \bar{y}_{...}\right)^2 = bn\sum_{i=1}^{a}\left(\bar{y}_{i..} - \bar{y}_{...}\right)^2 + an\sum_{j=1}^{b}\left(\bar{y}_{.j.} - \bar{y}_{...}\right)^2 + n\sum_{i=1}^{a}\sum_{j=1}^{b}\left(\bar{y}_{ij.} - \bar{y}_{i..} - \bar{y}_{.j.} + \bar{y}_{...}\right)^2$$
$$+$$
$$\sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k=1}^{n}\left(y_{ijk} - \bar{y}_{ij.}\right)^2$$

or symbolically we have

$$SS_T = SS_A + SS_B + SS_{AB} + SS_E$$

The ANOVA table is given in table 4.4

Table 4.4: Analysis of variance of Factorial completely randomized blocks

Source of variation	df	Sum of squares definition	Mean squares	Test statistic
A Treatments B Treatments Interaction Error	a-1 b-1 (a-1)(b-1) ab(n-1)	SSA SSB SSAB SSE	$MS_A = \frac{SS_A}{a-1}$ $MS_B = \frac{SS_B}{b-1}$ $MS_{AB} = \frac{SS_{AB}}{(a-1)(b-1)}$ $MS_E = \frac{SS_E}{ab(n-1)}$	$\frac{MS_A}{MS_E}$ $\frac{MS_B}{MS_E}$ $\frac{MS_{AB}}{MS_E}$ -
Total	abn-1	SST	..	..

Example 2

An engineer suspects that the surface finish of a metal part is influenced by the feed rate and the depth of cut. He selected three feed rates and randomly chooses four depths of cuts. He conducted an

experiment test the assertion above. The result obtained is given below in table 4.5 . Check whether the assertion is true.

Table 4.5: The result of ANOVA

Feed Rate	Depth			
	0.15	0.18	0.20	0.25
20 25 30	74, 64, 60 92,86,88 99,98,102	79,68,73 98,104,88 104,99,95	82,88,92 99,108,95 108,110,99	99,104,96 104,110,99 114,111,107

Tests of Between-Subjects Effects

Dependent Variable: value

Source	Type III Sum of Squares	df	Mean Square	F	Sig.
Model	326198.667[a]	12	27183.222	946.418	.000
feedrate	3160.500	2	1580.250	55.018	.000
depth	2125.111	3	708.370	24.663	.000
feedrate * depth	557.056	6	92.843	3.232	.018
Error	689.333	24	28.722		
Total	326888.000	36			

a. R Squared = .998 (Adjusted R Squared = .997)

From the table above we can easily see that the feed rate , the depth and the interaction effects are highly significant.

4.2.3 LATIN SQUARE DESIGN

We introduced the randomized complete block design as a design that reduces the residual error in an experiment by removing variability due to a known and controllable nuisance variable. While Latin-square designs are appropriate when two sources of extraneous variation are to be controlled. An extension to Graeco-latin-square designs allows the simultaneous control of three sources of extraneous variation .The Latin square design are ordinarily used when there is one factor of interest in the experiment and the experimenter desires to control two sources of variation.

CONSTRUCTION OF THE LATIN SQUARE DESIGN

1. Set up a table having k rows and k columns

2. Assign the letters A, B, . . . K to the cells in the first row of the table.
3. For the second row through the Kth of the table , place the first letter of the previous row in the last position and shift all other letters forward one position.
4. Randomly assign one of the blocking factor to the row and the other to the columns and assign the design factor to the body of the table.
5. Randomly assign the levels of the three factors to the row positions, column , positions and letters respectively.

THE STATISTICAL MODEL FOR A LATIN SQUARE DESIGN

The statistical model for a Latin Square Design is given as

$$Y_{ijk} = \mu + \alpha_i + \tau_j + \beta_k + \varepsilon_{ijk} \begin{cases} i = 1, 2, 3, \ldots p \\ j = 1, 2, 3, \ldots p \\ k = 1, 2, 3, \ldots p \end{cases} \tag{4.2}$$

where y_{ijk} is the observation in the i^{th} row and k^{th} column for the j^{th} treatment . is the overall mean α_i is the i^{th} row effect, τ_j treatment effect, β_k is the k^{th} column effect and ε_{ijk} is the random error..

The ANOVA for the Latin square consists of partitioning the total SS's of the $N = P^2$ observations into components for row, columns treatment and error for example

$$SS_T = SS_{Row} + SS_{Column} + SS_{Trearment} + SS_{Error}$$

with respective degrees of freedom

$$P^2 - 1 = P - 1 + P - 1 + P - 1 + (P - 1)(P - 1)$$

Example 3

The effect of five different catalyst (A, B, C, D. E) on the reaction time of a chemical process is studied. Each batch of new material is only large enough to permit five runs o be made. Furthermore, each run requires approximately one and half hours, so that only five runs can be made in one day. The experimenter decides to run the experiment as a Latin square so that day and batch effects may be systematically controlled. He obtains the data shown below in Table 3

Batch	Day

Table 4.6: ANOVA for Table: Analysis of variance of Latin Square

Source of variation	df	Sum of squares definition	Mean squares	Test statistic
Treatments	P-1	$SS_T = \frac{1}{P}\sum_{j=1}^{P} y_{.j.}^2 - \frac{y_{...}^2}{N}$	$MS_T = \frac{SS_T}{P-1}$	$\frac{MS_T}{MS_E}$
Row	P-1	$SS_R = \frac{1}{P}\sum_{i=1}^{P} y_{i..}^2 - \frac{y_{...}^2}{N}$	$MS_R = \frac{SS_R}{P-1}$	$\frac{MS_R}{MS_E}$
Column	(P-1)	$SS_C = \frac{1}{P}\sum_{k=1}^{P} y_{..k}^2 - \frac{y_{...}^2}{N}$	$MS_C = \frac{SS_C}{(P-1)}$	$\frac{MS_C}{MS_E}$
Error	(P-1)(P2)	SS_E obtained by subtraction	$MS_E = \frac{SS_E}{(P-2)(P-1)}$	-
Total	P^2-1	$SS_T = \sum_{i=1}^{P}\sum_{j=1}^{P}\sum_{k=1}^{P} y_{ijk}^2 - \frac{y_{...}^2}{N}$	..	..

	1	2	3	4	5
1	A=-1	B=-5	C =-6	D= -1	E =-1
2	B= -8	C=-1	D =5	E = 2	A = 11
3	C = -7	D=13	E= 1	A = 2	B = -4
4	D = 1	E = 6	A= 1	B =-2	C = -3
5	E = -3	A= 5	B = -5	C =4	D = 6

The model for the Latin Square experiment is given as

$$y_{ij(k)} = \mu + \alpha_i + \tau_j + \beta_k + \varepsilon_{ij(k)} \tag{4.3}$$

$i = 1, 2, 3, ..., p,\ j = 1, 2, 3, ..., p,\ k = 1, 2, 3, ..., p,$

y_{ijk} = the observation in ith row and jth column receiving the kth treatment

μ = overall mean

τ_k = the effect of the k treatment

α_i = the effect of the ith row

β_j = he effect of the jth column

Tests of Between-Subjects Effects

Dependent Variable: VALUE

Source	Type III Sum of Squares	df	Mean Square	F	Sig.
Model	552.000[a]	13	42.462	3.981	.011
row	68.000	4	17.000	1.594	.239
COLUMN	150.000	4	37.500	3.516	.040
TREATMENT	330.000	4	82.500	7.734	.003
Error	128.000	12	10.667		
Total	680.000	25			

a. R Squared = .812 (Adjusted R Squared = .608)

4.2.4 Graeco Latin Square Design

Consider a P x P Latin Square and super impose on it a second P x P Latin square in which the treatments are denoted by Greek letters. If the two squares, when super imposed, have the property that each Greek letter appears once and only once with each Latin letter, the two Latin squares are said to be orthogonal and the design obtained is called Graeco – Latin square. An example of a 5 x 5 Graeco- Latin square is shown in Table 4 .

The Graeco-Latin square design can be used to systematically control three sources of extraneous variability; that is, to block in three directions. The design allows investigation of four factors (rows, columns, Latin letters and Greek letters), each at P levels in only p2 runs. Greeco-Latin squares exists for all $p \geq 3$ except for p = 6

The statistical model for the Greeco-Latin squares design is

$$Y_{ijkl} = \mu + \theta_i + \tau_j + \omega_k + \phi_l + \varepsilon_{ijkl} \begin{cases} i = 1, 2, 3, \ldots p \\ j = 1, 2, 3, \ldots p \\ k = 1, 2, 3, \ldots p \\ l = 1, 2, 3, \ldots p \end{cases} \tag{4.4}$$

where y_{ijkl} is the observation in the i row and l column for Latin letter j and Greek letter k, μ is the overall mean , θ_i is the effect of the ith row, τ_j is the effect of Latin letter treatment j, ω_k is the

effect of Greek letter treatment k, Ψ_l is the effect of column l and $\square_{ijkl}$ is a Normally independent identically distributed random error with mean zero and variance $\sigma^2[NID(0, \sigma^2)] \ldots$

Table 4.7: 5x5 Graeco-Latin Square Design

	Column				
Row	1	2	3	4	5
1	Aα	Bγ	Cζ	Dβ	Eδ
2	Bβ	Cδ	Dα	Eγ	Aζ
3	Cγ	Dζ	Eβ	Aδ	Bα
4	Dδ	Eα	Aγ	Bζ	Cβ
5	Eζ	Aβ	Bδ	Cα	Dγ

The analysis of variance is very similar to that of a Latin square . Since the Greek letters appear exactly once in each row and column and exactly once with each Latin letter, the factor represented by the Greek letter is orthogonal to rows, columns and Latin letter treatments.. The general ANOVA table for the above design is given in Table 4.5

Example 4

The yield of a chemical process was measured using five batches of raw material, five acid concentrations, five standing times (A, B, C, D, E), and five catalyst concentrations $(\alpha, \beta, \delta, \gamma, \zeta)$. The Graeco-Latin square shown below was used

Table 4.8: Analysis of variance for Graeco-Latin Square Design

	Acid concentration				
Batch	1	2	3	4	5
1	Aα = 20	Bγ = 16	Cζ = 19	Dβ = 16	Eδ = 13
2	Bβ = 18	Cδ = 21	Dα = 18	Eγ = 11	Aζ = 21
3	Cγ = 20	Dζ = 12	Eβ = 16	Aδ = 25	Bα = 13
4	Dδ = 15	Eα = 15	Aγ = 22	Bζ = 14	Cβ = 17
5	Eζ = 10	Aβ = 24	Bδ = 17	Cα = 17	Dγ = 14

Analysis of Variance for Graeco Latin Square Design

Source of variation	df	Sum of squares definition	Mean squares	Test statistic
Latin letter Treatments	P-1	$SS_L = \frac{1}{P}\sum_{j=1}^{P} y_{.j.}^2 - \frac{y_{....}^2}{N}$	$MS_T = \frac{SS_T}{P-1}$	$\frac{MS_T}{MS_E}$
Greek letter Treatments	P-1	$SS_G = \frac{1}{P}\sum_{j=1}^{P} y_{..k.}^2 - \frac{y_{....}^2}{N}$	$MS_R = \frac{SS_G}{P-1}$	$\frac{MS_G}{MS_E}$
Row	(P-1)	$SS_R = \frac{1}{P}\sum_{i=1}^{P} y_{i...}^2 - \frac{y_{....}^2}{N}$	$MS_C = \frac{SS_R}{(P-1)}$	$\frac{MS_C}{MS_E}$
Column	(P-1)	$SS_C = \frac{1}{P}\sum_{k=1}^{P} y_{...\ell}^2 - \frac{y_{....}^2}{N}$	$MS_E = \frac{SS_E}{(P-2)(P-1)}$	-
Error	(p-3)(p-1)	SS_E obtained by subtraction		
Total	P^2-1	$SS_T = \sum_{i=1}^{P}\sum_{j=1}^{P}\sum_{k}^{p}\sum_{l=1}^{P} y_{ijk\ell}^2 - \frac{y_{....}^2}{N}$	..	..

From the ANOVA Table 13 above , we can see that the five batches of raw material, Acid concentration and Catalyst ,effects are not significant at 0.05 level while the standing times are significant.

4.2.5 NESTED OR HIERARCHICAL DESIGN

We shall conclude this section of our discussion by introducing the Nested or Hierarchical design..

A design in which the levels of one factor (e.g. factor A) is similar but not identical to each other at different levels of another factor (B) is called a nested or hierarchical design. Here the levels of factor A is nested under factor B. For example consider an experiment that is to be conducted using three samples of raw material from two vendors as shown in figure 4.1

Table 4.9: Tests of Between-Subjects Effects

```
Factor  Levels Values
C1   5   1   2   3   4   5
C2   5   1   2   3   4   5
C3   5   1   2   3   4   5
C4   5   1   2   3   4   5

Analysis of Variance for C5
Source   DF   Seq SS   Adj SS   Adj MS   F   P
C1   4   22.960   22.960   5.740   1.44   0.306
C2   4   6.160   6.160   1.540   0.39   0.813
C3   4   271.760   271.760   67.940   17.03   0.001
C4   4   32.160   32.160   8.040   2.02   0.185
Error   8   31.920   31.920   3.990
Total   24   364.960
a   R Squared = .996 (Adjusted R Squared = .987)
```

In such an experiment there is no physical or fundamental relationship between the samples 1, 2, and 3 from each vendor. In the language of design of experiment , the factor "samples would be said to be nested within the "vendor".

Construction of Hierarchically Nested Designs

The following steps are used in the construction of a hierarchical nested design

1. List the factors to be included in the experiment
2. Determine the hierarchy of the factors
3. Select randomly if possible, an equal number of levels for each factor within the levels of the preceding factor
4. Randomize the run order or the assignment of factor-level combinations to experimental units

The allocation of experimental resources in hierarchically nested designs yields more information on factors that are lower in the hierarchy than on those that are higher. For example, only two factors are studied in the design of figure 4, whereas twelve ingots are included in the design. In some circumstances it may not be desirable to have many factor levels at lower stages of the hierarchy. In such circumstance, unbalanced nested designs can be used to reduce the number of factor levels.

We give an example which we are going to use a soft ware called Minitab to analyze.

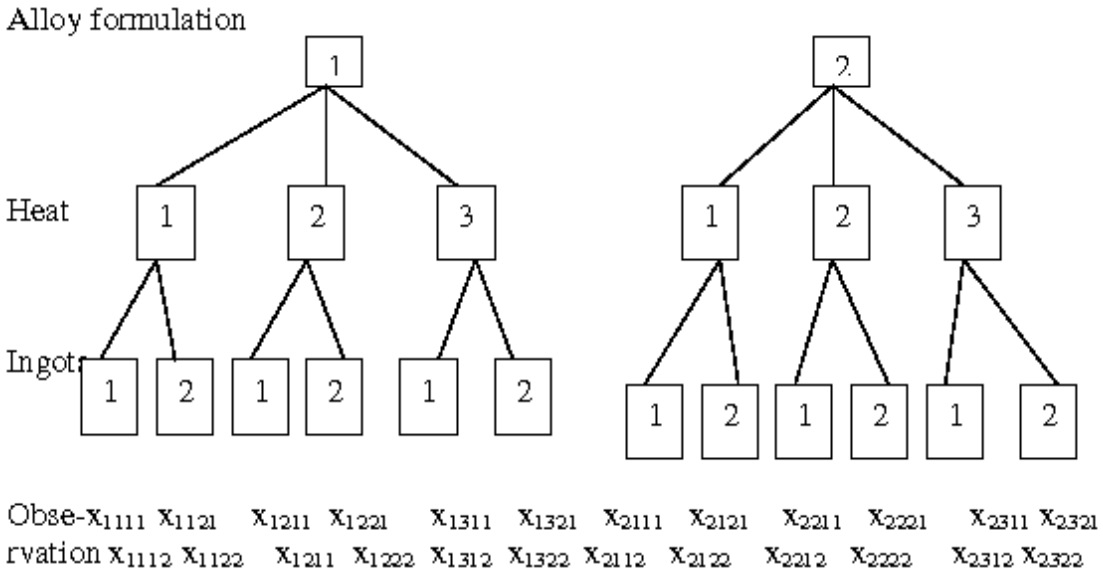


Figure 4.1: Illustration of nested design

Example 5

. An industrial engineer is studying the assembly of an electronic component to improve the speed of the assembly operator. He has designed three assembly fixture and two workplace layouts that seem promising. Operators are requires to perform the assembly, and it is decided to randomly select four operators for each fixture-layout combination. However, because the workplaces are different locations within the plant it is difficult to use the same four operators for each layout. Therefore, the four operators chosen for layout 1 are different individuals from the four operators chosen for layout 2. The treatment combinations for the design are ran in random order and two replications are obtained. The assembly units are measured in seconds and are shown in Table 4.10.

Table 4.10:

	Layout 1					Layout 2			
Operator	1	2	3	4		1	2	3	4
Fixture 1	22	23	28	25		26	27	28	24
	24	24	29	23		28	25	25	23
Fixture 2	30	29	30	27		29	30	24	28
	27	28	32	25		28	27	23	30
Fixture 3	25	24	27	26		27	26	24	28
	21	11	25	23		25	24	27	27

The model for the example on hierarchical experimental design given in example is

$$x_{ijkl} = \mu + \tau_i + \beta_j + \zeta_{k(j)} + (\tau\beta)_{ij} + (\tau\zeta)_{ik(j)} + \varepsilon_{(ijk)l} \begin{cases} i = 1, 2, 3, \ldots p \\ j = 1, 2, 3, \ldots p \\ k = 1, 2, 3, \ldots p \\ l = 1, 2, 3, \ldots p \end{cases} \tag{4.5}$$

$$SS_T = SS_F + SS_L + SS_{FL} + SS_{O(L)} + SS_{FO(L)} + SS_E$$

$$df : abcn - 1 = (a - 1) + (c - 1) + (b - 1)a + (a - 1)(c - 1) + (c - 1)(b - 1)a + abc(n - 1)$$

Expected Mean Square in the three stage Nested design where are the treatment factors are fixed

From the ANOVA table in tables 4.11 to 4.13, we observe that fixtures and Operations nested within layouts are significant at 1% level of significance while joint effects of operators and fixtures nested in layout is significant at 0.05 level of significance.

Table 4.11: Analysis of variance (ANOVA)for Nested Design

Source of variation	df	Sum of squares definition	Mean squares	Test statistic
τ_i	2	$SS_L = \sum_{j=1}^{a} \frac{y_{i\ldots}^2}{bcn} - \frac{y_{\ldots}^2}{abcn}$	$MS_T = SS_\tau$	$\frac{MS_T}{MS_E}$
β_j	1	$SS_{Layout} = \sum_{j=1}^{P} \frac{y_{\ldots j.}^2}{acn} - \frac{y_{\ldots}^2}{abcn}$	$MS_\beta = \frac{SS_\beta}{3}$	$\frac{MS_L}{MS_E}$
$\gamma_{k(\ell)}$	6	$SS_{FL} = \sum_{i=1}^{3}\sum_{j=1}^{2} \frac{y_{ij\ldots}^2}{cn} - \frac{y_{\ldots}^2}{abcn} - SS_{Fix} - SS_{lay}$	$MS_{O(L)} = \frac{SS_{O(L)}}{6}$	$\frac{MS_{FL}}{MS_E}$
$(\tau\gamma)_{(ik)\ell}$	12	$SS_{FO(L)} = \left\{ \sum_{i=1}^{2}\sum_{j=1}^{4}\sum_{k=1}^{3} \frac{y_{ijk.}^2}{n} - \sum_{j=1}^{4}\sum_{k=1}^{3} \frac{y_{.jk.}^2}{an} - \sum_{i=1}^{2}\sum_{j=1}^{4} \frac{y_{ij\ldots}^2}{cn} + \sum_{j=1}^{4} \frac{y_{j\ldots}^2}{acn} \right.$	$MS_E = \frac{SS_{FO(L)}}{12}$	$\frac{MS_{FO(L)}}{MS_E}$
Error	24	SS_E obtained by subtraction	$MS_E = \frac{SS_E}{24}$	
Total	$abcn-1$	$SS_T = \sum_{i=1}^{a}\sum_{j=1}^{b}\sum_{k}^{c}\sum_{l=1}^{n} y_{ijk\ell}^2 - \frac{y_{\ldots}^2}{abcn}$	..	..

Table 4.12: Analysis of Variance (Balanced Designs)

```
Factor  Type Levels Values
C3  fixed  3  1  2  3
C1  fixed  2  1  2
C2(C1)  fixed  4  1  2  3  4

Analysis of Variance for C5
Source  DF  SS  MS  F  P
C3  2  82.792  41.396  17.74  0.000
C1  1  4.083  4.083  1.75  0.198
C2(C1)  6  71.917  11.986  5.14  0.002
C3*C1  2  19.042  9.521  4.08  0.030
C3*C2(C1)  12  65.833  5.486  2.35  0.036
Error  24  56.000  2.333
Total  47  299.667
```

Table 4.13: Source Variance Error Expected Mean Square

```
component  term (using unrestricted model)
1 C3  6  (6) + Q[1,4,5]
2 C1  6  (6) + Q[2,3,4,5]
3 C2(C1)  6  (6) + Q[3,5]
4 C3*C1  6  (6) + Q[4,5]
5 C3*C2(C1)  6  (6) + Q[5]
6 Error  2.333  (6)
```

4.2.6 Split Plot Design

What Is a Split-Plot Design?

In simple terms, a split-plot experiment is a blocked experiment, where the blocks themselves serve as experimental units for a subset of the factors. Thus, there are two levels of experimental units. The blocks are referred to as whole plots, while the experimental units within blocks are called split plots, split units, or subplots. Corresponding to the two levels of experimental units are two levels of randomization. One randomization is conducted to determine the assignment of block-level treatments to whole plots. Then, as always in a blocked experiment, a randomization of treatments to split-plot experimental units occurs within each

Split-plot designs were originally developed by Fisher (1925) for use in agricultural experiments. As a simple illustration, consider a study of the effects of two irrigation methods (factor A) and two fertilizers (factor B) on yield of a crop, using four available fields as experimental units. In this investigation, it is not possible to apply different irrigation methods (factor A) in areas smaller than a field, although different fertilizer types (factor B) could be applied in

relatively small areas. For example, if we subdivide each whole plot (field) into two split plots, each of the two fertilizer types can be applied once within each

whole plot, as shown in Figure 1. In this split-plot design, a first randomization assigns the two irrigation types to the four fields (whole plots);then within each field, a separate randomization is conducted to assign the two fertilizer types to the two split plots within each field.

In industrial experiments, factors are often differentiated with respect to the ease with which they can be changed from experimental run to experimental run. This may be due to the fact that a particular treatment is expensive or time-consuming to change, or it may be due to the fact that the experiment is to be run in large batches and the batches can be subdivided later for additional treatments.

Why Use Split-Plot Designs?

In this paper, we advocate greater consideration of split-plot experiments for three reasons: cost, efficiency, and validity. In this section, we discuss each of these advantages in turn.

Cost The cost of running a set of treatments in split plot order is generally less than the cost of the same experiment when completely randomized. As Ganju and Lucas (1997, 1998, 2005) have noted, a properly implemented completely randomized design (CRD) requires that all factors must be independently reset with each run. If a factor level does not change from one run to the next and the factor level is not reset (e.g., by changing its level and then changing it back), inadvertent split plotting occurs. Such designs, termed random run order (RRO) designs by Ganju and Lucas (1997) are widely used and always analyzed inappropriately. We consider this issue in greater detail, under "Validity" later, and discuss here the relative cost of split-plot

experiments and true CRDs. Simple accounting tells us that the cost of a split-plot experiment can be significantly lessmbecause much of the cost (or time required) of running a split-plot experiment is tied to changes in the hard-to-change factors. If the cost of setting a hard to-change factor is Cw and the cost of setting an easy-to-change factor is Cs, and if there are nw whole Plots and ns splitplots,the CRD cost is ns (Cw+Cs), where as the SPD cost is nw Cw+nsCs,sothattheadditional cost for the CRD is (nsnw)Cw. Split-plot designs are useful precisely because this additional cost can be substantial. Anbari and Lucas (1994,2008) contrast the costs of CRDs and SPDs for various design scenarios. Bisgaard (2000) also discussed cost functions.

Efficiency Split-plot experiments are not just less expensive to run than completely randomized experiments; they are often more efficient statistically. Bisgaard (2000) show that, with a single hard-to-change factororasingly-to-change factor, use of a split plot layout leads to increased precision in the estimates for all factor effects except for whole-plot main effects. Overall measures of design efficiency have been considered by Anbari and Lucas (1994) and Goos and Vandebroek (2004). Considering two-level, full factorial designs only, Anbari and Lucas (1994) showed that the maximum variance of prediction over a cuboidal design region for a completely randomized design in some instances is greater than that for a blocked split-plot design involving one hard to-change factor. In other words, the G-efficiency of the split-plot design can, at times, be higher than the corresponding CRD. Similarly, Goos and Vandebroek(2001, 2004) demonstrated that the determinant of a D-optimal split-plot design (discussed later) frequently exceeds that of the corresponding D-optimal completely randomized design. These results indicate that running split-plot designs can represent a win–win proposition: less cost with greater overall precision. As a result, many authors recommend the routine use of split-plot layouts, even when a CR

Validity Completely randomized designs are prescribed frequently in industry, but are typically not run as such in the presence of hard-to-change factors. One of two shortcuts may be taken by the experimenter in the interest of saving time or money

4.2.7 How to Choose a Split-Plot Design

Choosing a “best” split-plot design for a given design scenario can be a daunting task, even for a professional statistician. Facilities for construction of split-plot designs are not as yet generally available in software packages (with SAS/JMP being one exception).Moreover, introductory textbooks frequently consider only very simple, restrictive situations. Unfortunately, in many cases, obtaining an appropriate design requires reading (and understanding) a relevant recent research paper or it requires access to software for generating the design. To make matters even more difficult, construction of split-plot experiments is an active area of research and differences in opinion about the value of the various recent advances exist.

The model for a split plot design is given as shown in equation 2.8

Example

An experiment is to compare the yield of three varieties of maize (factor A with a=3 levels) and four different levels of manure (factor B with b=4 levels). Suppose 6 farmers agree to participate in the experiment and each will designate a farm field for the experiment (blocking factor with s=6 levels). Since it is easier to plant a variety of oat in a large field, the experimenter uses a split-plot design as follows:

1. To divide each block into three equal sized plots (whole plots), and each plot is assigned a variety of oat according to a randomized block design.
2. Each whole plot is divided into 4 plots (split-plots) and the four levels of manure are randomly assigned to the 4 split plots.

A model for such a split-plot design is

$$Y_{hij} = \mu + \theta_h + \alpha_i + \varepsilon_{i(h)} + \beta_j + (\alpha\beta)_{ij} + \varepsilon_{hi} \tag{4.6}$$

Where

$$\begin{aligned} \varepsilon^w_{i(h)} i.i.di.id : N(0, \sigma^2_w), \\ \varepsilon^s_{j(hi)} i, i, d :: N(0, \sigma^2_s) \end{aligned} \tag{4.7}$$

$$\varepsilon^w_{i(h)} i.i.di.i.dN(0, \sigma^2_w), \varepsilon^s_{j(hi)} i, i, dN(0, \sigma^2_s) \tag{4.8}$$

are mutually independent, h=1, 2, ..., s, i=1, 2, ..., a, j=1, 2, ..., b.

Note the nested blocking structure: whole plots are nested within the blocks, and split-plots are nested within the whole plots. Two kinds of errors: $\varepsilon^w_{i(h)}$ representing the random effects of the whole plots, and $\varepsilon^s_{j(hi)}$ representing the random effects of split plots and random noises.

Note for testing of equal effects of factor A, the whole-plot mean square error is used. It should also be used for main effect contrasts of factor A. For tests or main effect contrasts of factor B, or AB interaction contrasts, the split-plot mean square error is used.

For main effects and interaction contrasts, the methods of multiple comparison of Bonferroni, Scheffe, Tukey, Dunnett, and Hsu can be used as usual.

Remark: If either levels of factor are assigned to whole plots as an incomplete block design, or the levels of factor B are assigned to split-plots as an incomplete design, the formulas of the sum of squares should be adjusted. But the degrees of freedom will remain the same. Estimates for main effects and interaction contrasts should be adjusted also.

Table 4.14: ANOVA Table for split-plot designs:

Source of Variation	D.F	Sum of Variation	Mean Square Error	Ratio
Block A Whole plot Error	s-1 a-1 (a-1)(s-1)	SS θ SSA SSEw	MSA MSEW	MSA/MSEW
B AB Split Plot Error	b-1 (a-1)(b-1) a(b-1)(s-1)	SSB SSAB SSES	MSB MSAB MSES	MSB/MSEs MSAB/MSEs
Total	n-1	Sstotal		

In general, within-whole-plot comparisons will generally be more precise than between-whole-plot comparisons. If the levels of all factors are easy to change, split-plot designs are recommended only when there is considerably less interest in one or more of the treatment factors.

Example

A rocket propellant manufacturer is studying the burning rate of propellant from three production processes. Four batches of propellant are randomly selected from the output of each process and three determinations of burning rate are made on each batch. The results follow. Analyze the data and draw conclusions.

	Process 1				Process 2				Process 3			
Batch	1	2	3	4	1	2	3	4	1	2	3	4
	25	19	15	15	19	23	18	35	14	35	38	25
	30	28	17	16	17	24	21	27	15	21	54	29
	26	20	14	13	14	21	17	25	20	24	50	33

4.3 Visualization

In this section we shall briefly consider the last part of the subject matter which is visualization which we defined in the section on introduction. This part of the work made a copious use of the article on Educause Learning Initiative (www..educause.edu/eli) October 2007

Table 4.15: Tests of Between-Subjects Effects

Dependent Variable:value1						
Source	Type III Sum of Squares	df	Mean Square	F	Sig.	
batch1	Hypothesis	230.306	3	76.769	.249	.859
	Error	1847.278	6	307.880a		
temp1	Hypothesis	22.889	2	11.444	.465	.658
	Error	98.444	4	24.611b		
Process1(temp1)	Hypothesis	98.444	4	24.611	1.332	.296
	Error	332.667	18	18.481c		
Process1(batch1)	Hypothesis	1847.278	6	307.880	16.659	.000
	Error	332.667	18	18.481c		
a. MS(Process1(batch1))						
b. MS(Process1(temp1))						
c. MS(Error)						

4.3.1 What is it?

Data visualization is the graphical representation of information. Bar charts, scatter graphs, and maps are examples of simple data visualizations that have been used for decades. Information technology combines the principles of visualization with powerful applications and large data sets to create sophisticated images and animations. A tag cloud, for instance, uses text size to indicate the relative frequency of use of a set of terms. In many cases, the data that feed a tag cloud come from thousands of Web pages, representing perhaps millions of users. All of this information is contained in a simple image that you can understand quickly and easily.

More complex visualizations sometimes generate animations that demonstrate how data change over time.

4.3.2 Who's doing it?

Data visualizations have long been used in academic settings, but many instructors are using new technologies to create rich, compelling visualizations to help students understand concepts more quickly and deeply. A history professor, for example, could use a visualization that shows which industries prospered and which suffered during the wars and economic cycles of the 20th century to explain demographic shifts and the social changes that followed. An economics professor might use the same

visualization to explain the financial connections in the national or global economy. Working with the visualization team at the Renaissance Computing Institute, a faculty member in the Department of Soil Science at North Carolina State University created an animated, interactive visualization that shows how fertilizer nitrates enter groundwater

4.3.3 How does it work?

Visualizations encompass a wide and growing range of projects, reflecting creative ways of representing all sorts of data visually, with virtually no limit to what kind of information can be translated into an image. Visualizations have been created that represent the possible moves on a chess board, the structure of a piece of music, and the messages in a person's e-mail inbox, to name a few. The designer of a visualization determines which visual element (color, shape, size, motion, and so forth) will represent individual data points. Images can be 2D or 3D, can be fixed or dynamic, and can allow user interaction. One application, for example, shows political contributions to various candidates. You can select a state, a political race in that state, and a monetary threshold for contributions. The application builds a 2D image that shows who supported each candidate and at what level (based on the relative sizes of the circles that represent contributors), revealing interesting webs of political influence. Astrophysicists use visualizations to create 3D images that model the forces of a supernova. In each case, images or animations are the products of applications that render data in a visual form based on the design of the visualization.

4.3.4 Why is it significant?

Computer systems generate and store massive and growing amounts of data. At the same time, advanced networks, distributed processing, and other developments allow unprecedented access to data. Data visualizations offer one way to harness this infrastructure to find trends and correlations that can lead to important discoveries. Representing large amounts of disparate information in a visual form often allows you to see patterns that would otherwise be buried in vast, unconnected data sets. As opposed to the traditional hypothesis-and-test method of inquiry, which relies on asking the right questions, data visualizations bring themes and ideas to the surface, where they can be easily discerned. Visualizations allow you to understand and process enormous amounts of information quickly because it is all represented in a single image or animation. Moreover, virtually any kind of data from a broad range of academic disciplines can be represented visually, making data visualization a potentially valuable approach to learning for a large number of students and researchers

4.3.5 What are the downsides?

Visualizations rely on accurate and matched data. If data are incomplete or faulty, or if data sets use different definitions or units, these issues must be resolved in order to create a valid visualization,

and this can be time-consuming. Even if the data are reliable and consistent, a poorly conceived visualization might show nothing of consequence or exaggerate the significance of certain trends, resulting in flawed or misleading conclusions. In some cases, a lot of time and trouble go into a visualization that adds nothing to an understanding of the data that you wouldn't find in a simple table or even a textual description. Finally, users who prefer conventional ways of learning and processing information might be uncomfortable working with data visualizations, which require a different approach to understanding data.

4.3.6 Where is it going?

As data visualization tools become more powerful and more widely available, users will create representations of the relationships between data not previously considered together. Some of these experiments will show interesting relationships that can lead to new understanding about correlations and causality between a wide range of variables. Data visualizations will take greater advantage of animation tools, allowing users to control the parameters of those animations, and visualization projects will increasingly be cross-disciplinary, allowing experts in diverse fields to seek and express relationships between their respective areas of inquiry. One project, for example, compares data about individuals' stress levels with their geographic location. Many mashups employ data visualization to represent hidden connections, such as images that show the complex networks of users on social networking sites. These and other applications will broaden the general sense of how data can be read and interpreted.

4.3.7 What are the implications for teaching and learning?

Many data visualization projects allow users to develop as well as interpret images. Involving students in the formulation of visualizations requires them to think in creative ways about the material at hand and how the information can be effectively communicated through shapes, colors, animations, or other visual elements. By examining data and deciding how they are presented, students might have a stronger sense of ownership of that data. As consumers of visualizations, students have the opportunity to break out of traditional modes of thinking about data and challenge themselves to ask critical questions about what data do and do not show and about relationships between sets of data. Visual literacy is an increasingly important skill, and data visualizations are another channel for students to develop their ability to process information visually. For students and researchers, visualizations can be an important data-analysis tool, uncovering key findings that would otherwise be difficult or impossible to perceive.

References

Altoveros E.C. Applied experimental designs for Agricultural Research (Unpublished lecture note)

Anbari and Lucas (1994) "Super-Efficient Designs: How to Run Your Experiment for Higher Efficiencies and Lower Cost". ASQC Technical Conference Transactions, pp. 852–863.

Bisgaard, S. (2000). "The Design and Analysis of $2k-p \times 2q-r$ Split Plot Experiments". Journal of Quality Technology 32, pp. 39–56.

Box, G.; Hunter, W.; and Hunter, S. (2005). Statistics for Experimenters: Design, Innovation, and Discovery,2nd edition. New York, NY: Wiley-Interscience.

Educause Learning Initiative (www..educause.edu/eli) October 2007

Fisher, R. A. (1925). Statistical Methods for Research Workers.Edinburgh: Oliver and Boyd.

Goos and Vandebroek (2001, 2004) Outperforming Completely Randomized Design: Journal of Quality Technology, 36 pp 12 26

Ganju, J. and Lucas, J. M. (1997). "Bias in Test Statistics when Restrictions in Randomization Are Caused by Factors".Communications in Statistics: Theory and Methods 26, pp. 47–63.

Isik F (2006) Design of Experiments and Data Handling (Unpublished Lecture note)

Lazic´(2004) Design of Experiments in Chemical Engineering a Practical guide. Wiley-Veh Verlag GmbH & co KGaH

Little T.M. and Hills (1972) Statistical Methods in Agricultural Research (Unpublished Teaching Manual)

Mason R.I., Gunst, R,F and Hess, J. L. (2003) Statistical Design of experiments with application in Engineering and Science 2nd edition Wiley Interscience New York

McGee, M.G. (1979) Human spatial abilities: Psychometric studies and environmental, genetic, hormonal and neurological influences. Psychological Bulletin 86, 889-918

Montgomery D G. Design and Analysis of Experiment 5Thedition John-Wiley Inc New York

Minitab soft wares

SPSS soft wares

Strong S and Smith R. (2002) Spatial Visualization: fundamentals and Trends in Engineering Graphics. Journal of industrial Technology Vol. 18, No 1

MATLAB/SCILAB Training Course, Vol. 1, pp. 66–72, 2011

Received 28 October 2011

CHAPTER 5

COMMONLY USED SOFTWARE FOR DATA ANALYSIS AND VISUALIZATION IN EXPERIMENTAL AGRICULTURE

by

BAIYERI, K. P.
Department of Crop Science,
University of Nigeria, Nsukka

5.1 Introduction

Historically, most statistical designs have its root in agricultural research; because most materials used for agricultural researches are often not homogenous in nature, it was very needful to utilize experimental designs that will limit interferences due to heterogeneity of experimental materials with experimental outputs. Sir R. A. Fisher and his colleagues at Rothamsted Experiment Station, UK were the pioneer developers of most of agricultural designs of experiment and analysis of variance techniques in use today (http://www.iasri.res.in/ebook/EBADAT/1-Computer/Usage and/Statistica/Software/Packages/1-Role_statistics-computers_in/Ag/Res.pdf, Oct. 25, 2011).

The backbone of experimental agriculture is research experimentation. Whenever there is the need to ascertain validity of any assertion, there will be a corresponding need to generate data and on the basis of data generated draw valid conclusions. Agricultural experimentations are conducted in the laboratories, greenhouses, glasshouses, screen-houses, animal houses and in the fields. Any agricultural experimentation has two major components, which are, designing the experiment (or the way of generating the data) and the analysis of the data generated to draw meaningful and valid conclusions.

In the earlier days of agricultural experimentation, designs were generated in such a way that there was ease in the data analysis. Such experimentation was merely explorative, requiring further detailed experimental methods for valid or predictable conclusions to be made. With the advent of high-speed computers, complex but more informative designs are easily utilized in agricultural experimentation. Availability of many statistical software packages has made analyzing agricultural data no longer a problem.

The basic experiment designs utilized in most agricultural and biological experimentation are Completely Randomized Design (CRD), Randomized Complete Block Design (RCBD), and Latin Square Design (LSD). When the purpose of the experimentation is to evaluate the effects of two or more factors, other models are superimposed on these basic designs. Two of such models that handle multi-factors experimentation are split-plot and factorial designed experiments. The models ensure that interactions among/between factors can be estimated.

Multivariate statistics is a form of statistics encompassing the simultaneous observation and analysis of more than one statistical variable (http://en.wikipedia.org/wiki/Multivariate_statistics). There are several multivariate statistics; however the most commonly utilized models will be discussed briefly:

Principal component analysis (PCA) is a mathematical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of uncorrelated variables called principal components. The number of principal components is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has as high a variance as possible (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it be orthogonal to (uncorrelated with) the preceding components (http://en.wikipedia.org/wiki/Principal_components_analysis, Oct.25, 2011). PCA) is commonly used for visualization of genetic relatedness of collections of plant or animal materials in agriculture.

Factor analysis is similar to PCA but allows the user to extract a specified number of synthetic variables, fewer than the original set, leaving the remaining unexplained variation as error (http://en.wikipedia.org/wiki/Principal_components_analysis, Oct.25, 2011). The extracted variables are known as latent variables or factors; each one may be supposed to account for covariation in a group of observed variables.

Additive Main Effect and Multiplicative Interaction (AMMI) model and Genotype-Genotype-Environment (GGE) model are biplot models for data visualization in multi-location yield trials and related experimentation. These models integrate both PCA and ANOVA to execute data analysis, thus output of these models are more informative than either PCA or ANOVA. For International Varietal Yield trials and/or modeling of genotype-by-environment interaction AMMI or GGE model is the first to be considered. GGE biplot analysis executes genotype-by-traits; this analysis permits scientist to visualize the relationships between sets of genotypes and series of morpho-physiological traits. The visual display enables scientist to make selection of genotype that possesses best combination of traits to meet targeted needs. The GGE biplot analysis for genotype-by-traits model can also pro-

duce a visual display that enables scientist to identify trait that discriminate among genotypes or that represents a general pattern among a set of genotypes (See Figures 1 and 2).

Important considerations that suffice in the choice of a design for a given experimental situation include:

- Cost effectiveness, keeping in view the scarce and expensive resources.
- Reliability/predictability/Precision of result output
- Robustness of the design- able to absorb various shocks like loss of data points, presence of outliers, interchange and/or exchange of treatments, model inadequacy, etc., besides providing as small an experimental error as possible, or in other words as small a coefficient of variation (CV) value as possible.

The purpose of this brief is to point out various statistical software packages commonly used for designing and analysis of experiments in agricultural and biological systems. A number of software packages available include Statistical Analysis System (SAS), Statistical Package for Social Sciences (SPSS), GENSTAT, MINITAB, MSTATC, MS-Excel, Biplot analytical software (like AMMI, GGE) etc. It may not be possible to describe detailed features of all these software packages right here; however, some that have been found to be more useful in agricultural and biological research are discussed.

5.2 SAS (Statistical Analysis System)

This is comprehensive software that deals with many problems related to statistical analysis, spread sheets, data creation, and graphics, etc. The three components of the SAS system are host, portable applications and data. Host provides all the required interfaces between the SAS system and the operating environment. Functionalities and applications reside in portable component and the user supplies the data. There are three basic sections of SAS program viz (1) DATA section, (2) CARDS section and (3) PROCEDURE section. The data section is invoked with the DATA statement. In the cards section, user supplies the data. SAS system has a collection of ready to use programs called PROCEDURES. These procedures can be invoked with the PROC statements.

Most of the analysis of data generated from designed experiments can be performed using PROC GLM. However, if one is quite sure that the data is balanced then one can use PROC ANOVA in place of PROC GLM. The PROC GLM or PROC ANOVA, CLASS and MODEL statement are must for this analysis. The CLASS statement defines the variables for classification. The MODEL statement names the dependent variables and independent variables or effects. If no effects were specified in the MODEL statement, ANOVA fits only the intercept. Included in the ANOVA output are F-tests of all effects in the MODEL statement. All of these F-tests use residual mean squares as the error term.

However, the residual mean square is not the appropriate term such as main-plot effects in split-plot experiment.

Then one can use the TEST statement for selecting the appropriate terms for testing the various effects. For example, one may select the rep * main plot treatments effects as error term in split plot experiments for testing the hypothesis regarding the main-plot effects. One may also obtain the means of the listed effects in the MODEL and use several multiple comparison procedure viz: Least Significant Difference, Duncan's New multiple-range test, Waller-Duncan test, Turkey's Honest Significant Difference.

However, if the MODEL statement includes more than one dependent variable like grain yield, straw yield, etc., an additional multivariate statistics can be requested with the MANOVA statement. There is also a provision of testing the effects of user-defined contrasts. This option makes the PROC GLM more versatile. Using this option one may analyze the experimental data from block designs for factorial experiments as general block design and various main effects and interaction effects can be obtained through contrasts analysis. It therefore gives a unified procedure for the analysis of factorial experiments with or without extra treatments, balanced or partially balanced, confounded, etc. One can perform the Analysis of Covariance (ANCOVA) to test the effects of covariates and covariate interactions with factors. One can also analyzed mixed effects models and models involving nested effects. One can also estimate the variance components using standard procedures of variance component estimation like maximum likelihood (ML), restricted maximum likelihood (REML), MIVQUE, or Henderson's method III.

Using SAS one can also fit a second order response surface in one go. One can get the co-ordinates of the stationary point, nature of the stationary point whether it is a point of maxima, minima or saddle point. One can also obtain the predicted response at the stationary point. The interesting feature of SAS is that it has its own IML (interactive matrix language), which includes many functions and subroutines. One can write own SAS code for specific applications.

5.3 Statistical Package for Social Sciences (SPSS)

SPSS is a very comprehensive and flexible package that consists of software tools for data entry, data management, statistical analysis and presentation of results. The data generated from the designed experiments can be very easily analyzed for both balanced and unbalanced data. Starting from the simplest designs, i.e. completely randomized design (CRD), randomized complete block designs (RCBD) to designs with nested factors, row-column designs, can be analyzed using SPSS.

The general Linear Model Univariate procedure provides analysis of variance for one dependent variable by one or more factors and /or variables. The factor variables divide the population into groups. Using this General Model procedure, one can test null hypotheses about the effects of other variables on the means of various groupings of a single dependent variable, investigate interactions between factors as well as the effects of individual factors, some of which may be random. In addition, the ef-

fects of covariates and covariate interactions with factors can be included. After an overall F-test has shown significance, post hoc tests can be used to evaluate differences among specific means. Further analysis using the 'contrasts' and 'post-hoc' options can be done in order to investigate the differences in the particular treatment effects.

5.4 GENSTAT

This was developed by Rothamsted Experimental Station, United Kingdom. It is an interactive general-purpose statistical package with flexible command language. It covers most of the standard statistical analysis procedures and can also be easily extended for newer techniques. It is particularly useful for the analysis of designed experiments, the fitting of linear models (including regression) and general linear models.

It also covers most multivariate techniques, time-series analysis and optimization. Under the analysis of designed experiments it covers the following:

- BLOCK STRUCTURE: defines the blocking structure
- COVARIATE: covariates for covariance analysis
- TREATMENT STRUCTURE: defines the treatment terms
- ANOVA: performs analysis of variance
- ADISPLAY: displays output produce by ANOVA
- AKEEP: copies ANOVA information into data structure
- REML: fits variance components model by residual maximum likelihood
- VCOMPONENTS: defines the model for REML
- VDISPLAY: displays further output from REML analysis
- VKEEP: copies results from REML into data structure. One can also utilize the various GENSTAT functions and libraries.

5.5 MSTAT-C

MSTAT is a powerful tool used to simplify and enhance field and laboratory research. Researchers can use this software to generate the design of experiments (RCB, CRD, Lattice), print labels, and

collect, organize, and analyze the data. MSTAT is a complete package containing all the programs listed below: Experiment design MSTAT has two experiment design programs. EXP Series will design experiments with from one to five factors and with up to four splits. It will also generate files for field books, labels, and maps and that can be edited, printed and/or stored. VAR Series will design experiments with one factor. A square or rectangular lattice design can be created if the number of factor levels is latticeable (two or three replicates) or a one factor RCB design. VAR Series will generate files for field books, labels and maps that can be edited, printed and/or stored. The labels can be of three types - planting, harvest, or sampling. DATA COLLECTION Data collected from the field or laboratory can be entered using MSTAT's editor. If MSTAT was used to generate the experiment design, the file to receive research data is in the same plot order as the collected data. Data can be entered through the screen editor or by file transfer from data collectors or other software (ASCII Compatible). Once data are keyed in, they can be sorted, transformed, or edited using MSTAT. DATA ANALYSIS MSTAT ANOVA programs include one/two way, factorial, lattice, nonorthogonal, hierarchical, nearest neighbor, and latin square. Other data analysis programs include T-test, LSD, Duncan's, Tukey's, Student-Neuman-Keuls, missing value estimator, orthogonal contrasts, summary statistics, principal components, regression, correlation, and multiple regression. DATA FILE CREATION, EDITING, AND SORTING There are many ways to enter, manipulate, and transform data. Data sets can be imported and exported in ASCII file format for easy transfer to and from other software (word processing, spreadsheet, and/or database). Data can be sorted (with up to 14 keys), transformed using equations, and manipulated to select specific types of data within a data set. A data set can be "printed" to a file, to screen or to a printer. DESCRIPTIVE STATISTICS AND PLOTTING MSTAT programs (STAT, MEAN, GROUPIT, FREQ, and TABLES) can be used to compute basic tables of statistics such as minimum, maximum, means, standard deviation, skewness, and kertosis. Data can be grouped into specific categories based on data values within a data set. Data frequency can be computed and histograms plotted. An x-y scattergram can be created and printed. MGRAPH is an additional routine that can be used alone or with MSTAT to create graphic output of data (line, pie, bar). PLANT BREEDING BR Series can be used to print books and labels for a plant breeding program. It will automatically keep track of parentage, generation, plot numbers, cross numbers and character index. AC Series is a complete data management series for breeding materials. This series can be used to create a master accession file, crossing block files, and F1 to F9 breeding files. Field trials can be generated as well as six book styles and five types of labels. The Master Accession file includes breeding material information (accession number, descriptors, name, source and pedigree) and can be updated. ADDITIONAL PROGRAMS MSTAT has programs for multivariate analysis (Hotellings's T squared statistic, multiple discriminant and principle component analysis, nonparametric statistics). ECON calculates the net benefit, risk, and marginal rate of return for experiments with treatments of varying costs. SEASONS is an economic/seasonal regression program used to analyse seasonal commodity prices. MSTAT is written in the C programming language and runs on DOS compatible machines with a hard drive (2 MB used by MSTAT) and a minimum of 512 K in RAM. MSTAT can be used under WIN 3.x, WIN 95, WIN 98, WIN XP, WIN NT, and WIN ME. MSTAT is available in English. MSTAT is a very easy program to use. The manual uses sample data

files to explain the operation of each program.

5.6 MS-EXCEL

Can be utilized for the analysis of experimental data; some of the inbuilt functions and procedures are given below: In the TOOLS menu, the option of Data Analysis can be used for carrying out Analysis of Variance (ANOVA) for single factor (one-way classified ANOVA), two factor with replications (Two Factor Completely Randomized design or two way classified data with m- observations per cell) and two factor without replication (Randomized Compete Block Design).

MS-EXCEL can be used for transformations in the analysis of experimental data. The following transformations can be done using the inbuilt functions given in the parenthesis: Arcsine (ASIN), logarithmic (LN) and Square root (SQRT). One can also do the matrix multiplication, matrix inverse and transpose of a matrix using the options as MMULT, MINVERSE and TRANSPOSE respectively. One can also write the MACROS separately using MACROS under the TOOLS menu or Visual Basic can be used for writing Macros.

References

Citations were made from the under-listed references

Anonymous (1999) . Genstat For Windows Introductory Manual. Statistical Services Centre, University of Reading, U.K.

Gelman, A. (2005) . Analysis of Variance: Why it is more important than ever (with discussion). Annals of Statistics 33, 1-53.

Malhotra, P.K. and Goyal, R.C. (2009) . Information and Software Development at Indian Agricultural Statistics Research Institute: An Overview. IASRI- An Era of Excellence, IASRI Golden Jubilee Publication, 161-184.

Mead, R; Curnow, R.N. and Hasted, A.M. (1993) . Statistical method in agriculture and experimental biology (2nd edition).

O'Neill, M.O. and Lee, C.J. (2009) . Statistical modeling of a split-block agricultural field experiment. StATS Ltd, Pty Australia and Agro-Tech Inc., USA.

Stern, R.D., Allan, E. and Coe, R (1999) Software familiarization. Lecture note (Part 2). Data analysis of agroforestry experiments. ICRAF

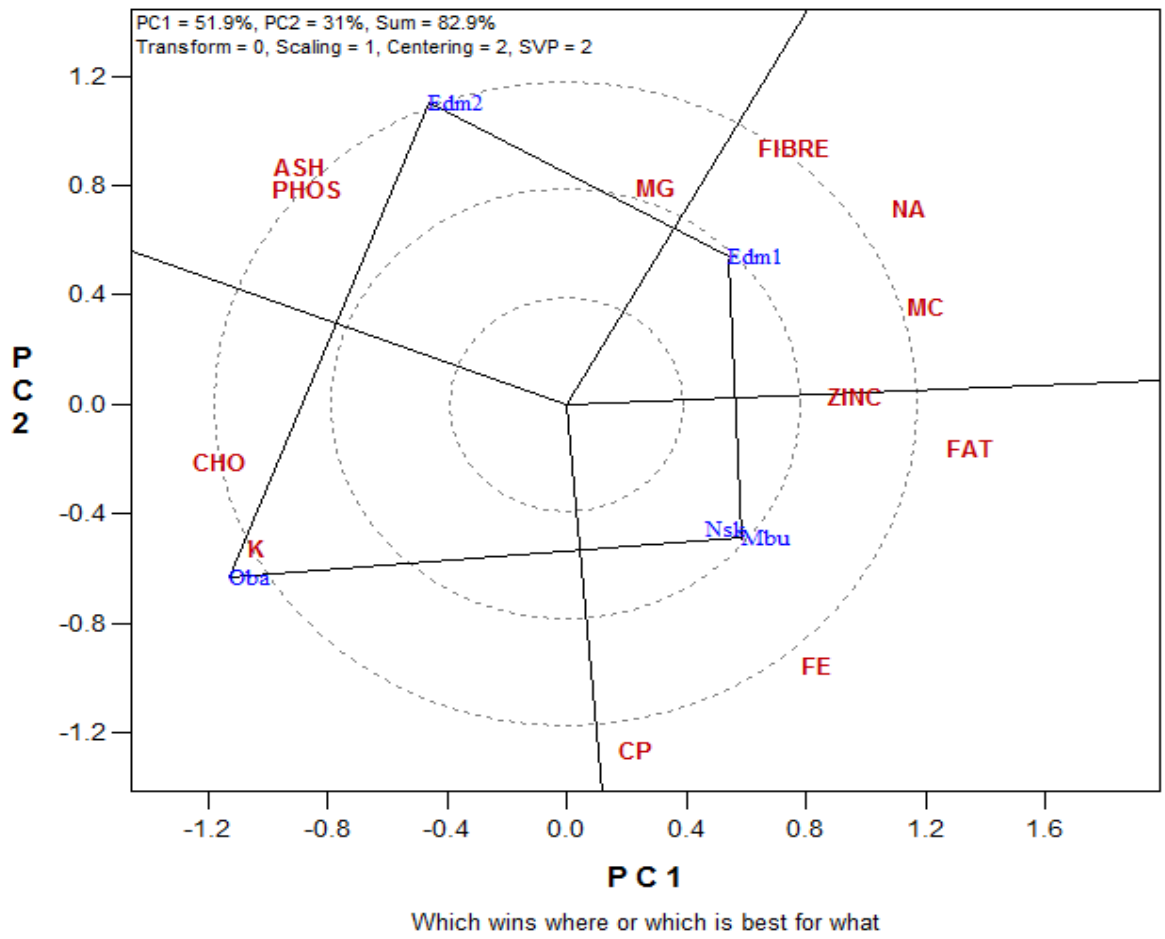


Figure 5.1: A GGE biplot analysis visualizing five accessions of African Walnut in relation to their nutritional qualities

Accession occupied different vertexes indicating genetic distinctiveness in relation to the nutritional traits evaluated

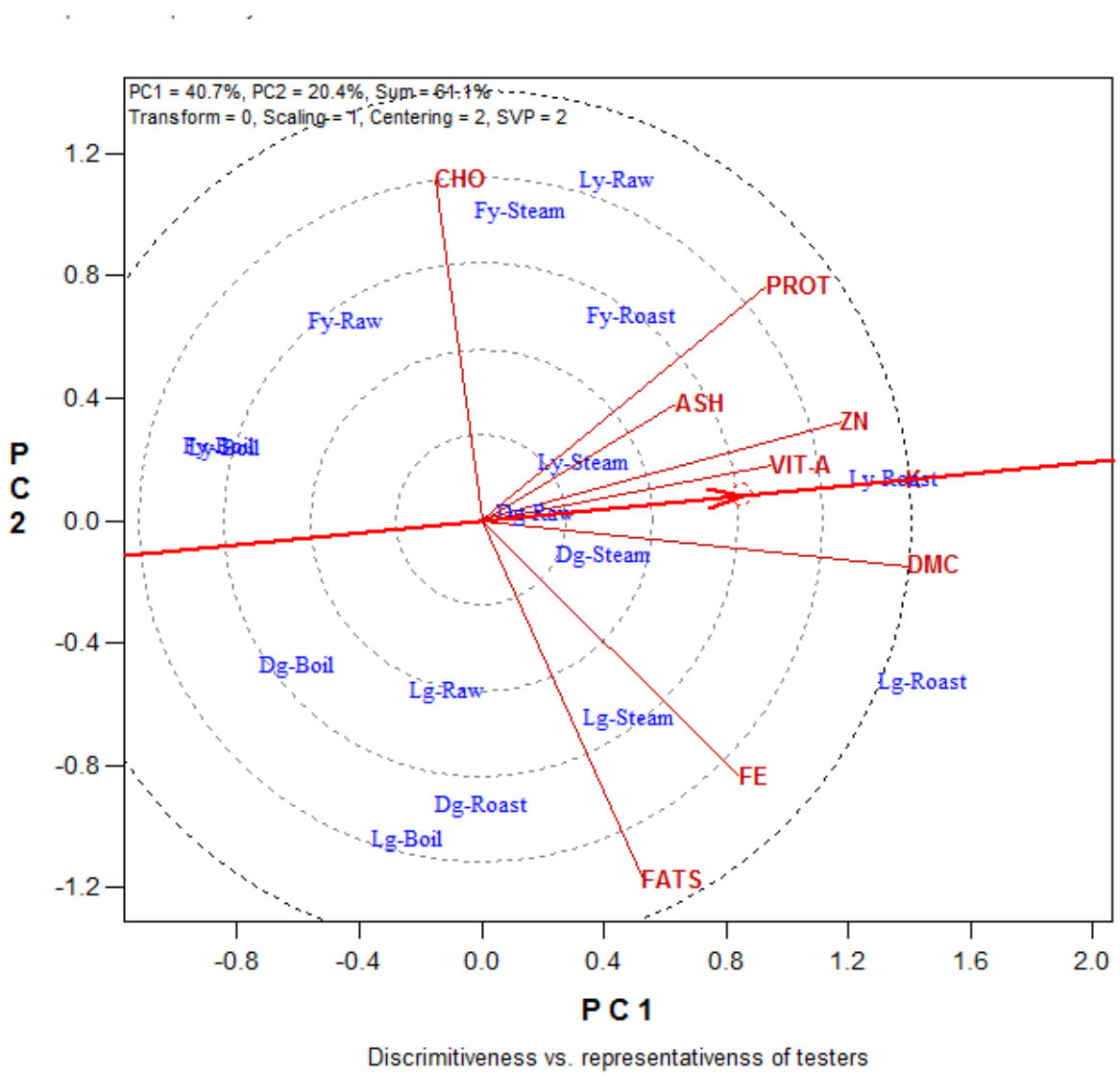


Figure 5.2: A GGE biplot analysis visualizing processing methods and ripening stages of plantain in relation to nutritional qualities of the fruit.

The biplot revealed fairly ripe fruit that was roasted was the most representative of the sixteen treatment combination, suggesting that this ripening stage and the processing method ensured the highest nutritional quality of plantain.

MATLAB/SCILAB Training Course, Vol. 1, pp. 75–81, 2011

Received 10 May 2011

CHAPTER 6

APPLICATION OF DATA ANALYSIS/DATA VISUALIZATION IN PHARMACEUTICAL RESEARCH AND DEVELOPMENT

by

OSADEBE, PATIENCE OGOAMAKA
PROFESSOR OF PHARMACEUTICAL & MEDICINAL CHEMISTRY
UNIVERSITY OF NIGERIA NSUKKA

6.1 Introduction

6.1.1 Overview of Analytical statistical tools

Maximum information are usually obtained from analytical results when statistical techniques are applied to their interpretation. Statistical treatment of experimental results guarantees some confidence limits and hence adequacy of obtained results. Introduction contd

Traditional Statistical tools that have been applied in pharmaceutical and biological fields are

- mean,
- median,
- Variance
- Standard deviation
- Standard error of mean Introduction

- Confidence limits or intervals
- Significance tests
  - Student's T-tests
  - F-tests or ANOVA
  - Chi-squared tests
  - Correlation and regression analysis

Statistical analysis of experimental data is a progressive process that starts with organization of numerical information and commonly ending with sophistcated calculations and interpretations that permit valid conclusions to be drawn and decisions to be made.

In recent times new and robust approaches to data analysis which includes the subject of visualization have emerged in the field of pharmaceutical research and Drug development. Nowadays, pre-clinical drug discovery relies on huge volumes of inter-related multivariate data which must be presented in interpretable form in order to make sense of them and be able to take quality decisions based on this plethora of information.

Intuitive visualization methods came into play as a result of the need to reduce the assumed complexity of data following reduction in its dimensionality.

6.1.2 Definition of Terms

Data mining: Refers to a series of computational techniques designed to sift through large volumes of data in order to identify important trends and make predictions

Visualization: This is the term used to describe any methodology that project data in lower-dimensional space usually two or three dimensions while maintaining large percentage of information from the higher dimensional space

Data Visualisation has been widely recognised as a tool to obtain information from large quantities of data. It is a popular way of presenting complex data in a simpler form

6.1.3 Data types in pharmacy research

These include

1. Biological data such as protein, DNA, RNA sequences etc
2. Chemical data such as small molecules (drugs)

3. Experimental data such as absorbance and pH readings.
4. Miscellaneous data such as demographic, disease incidence

6.2 Visualization methods

- Generative topographic mapping(GTM)
- Hierarchial GTM
- Principled component Analysis(PCA)
- Sammon's Mapping
- Self-organising Maps(SOMs)

6.2.1 Visualization tools for biological data

- VMD (`http://www.ks.uiuc.edu/Research/vmd/`)
- PyMol (www.pymol.org)
- Maestro (`https://www.schrodinger.com/products/14/12/`), etc

6.2.2 Visualization tools for chemical data

- Maestro (https://www.schrodinger.com/products/14/12/)
- Avagadro (avogadro.openmolecules.net)
- Chemcraft (www.chemcraftprog.com) etc
- Ligand explorer (`http://www.rcsb.org/pdb/explore/viewerLaunch.do?viewerType=LX&structureId=1Y7H&hetId=SCN`)

6.2.3 Data analysis in pharmaceutical research and development

- Excel
- SPSS
- SmartDraw

6.2.4 Visual molecular dynamics (VMD)

VDM Is a molecular modelling and visualization computer program. VMD is primarily developed as a tool for viewing and analyzing the results of molecular dynamics simulations, But it also includes tools for working with volumetric data, sequence data, and arbitrary graphics objects. Molecular scenes can be exported to external rendering tools such as POV-Ray, Renderman, Tachyon, VRML, and many others. Users can run their own Tcl and Python scripts within VMD as it includes embedded Tcl and Python interpreters. VMD is available free of charge, and includes source code, but it is under a non-free license (Wikipedia)

VMD is designed for modeling, visualization, and analysis of biological systems such as proteins, nucleic acids, lipid bilayer assemblies, etc. It may be used to view more general. molecules, as VMD can read standard Protein Data Bank (PDB) files and display the contained structure. VMD provides a wide variety of methods for rendering and coloring a molecule: simple points and lines, CPK

Spheres and cylinders, licorice bonds, backbone tubes and ribbons, cartoon drawings, and others. VMD can be used to animate and analyze the trajectory of a molecular dynamics (MD) simulation. In particular, VMD can act as a graphical front end for an external MD program by displaying and animating a molecule undergoing simulation on a remote computer.

Key features of VMD include:

- Support for all major computer platforms
- Support for multicore processors
- Support for GPU accelerated computation

  Many excellent VMD tutorials developed locally, and by the research community at large No limits on the number of molecules, atoms, residues or number of trajectory frames, except available memory
- Many molecular rendering and coloring methods
- Stereo display capability
- Extensive atom selection syntax for choosing subsets of atoms for display (includes boolean operators, regular expressions, and more)
- Support for over 60 molecular file formats and data types through an extensive library of built-in file reader/writer plugins and translators
- VMD includes a multiple sequence alignment plugin, a unified bioinformatics analysis environment that allows one to organize, display, and analyze both sequence and structure data for proteins and nucleic acids.

- Ability to export displayed graphics to files which may be processed by a number of popular ray tracing and image rendering packages, including POV- Ray, Rayshade, Raster3D, and Tachyon.
- User-extensible graphical and text-based user interfaces, built-on standard Tcl /Tk and Python scripting languages
- Extensions to the Tcl language which enable researchers to write their own routines for molecular analysis
- Modular, extensible source code using an object-oriented design in C++, with a programmers guide describing the program architecture and source code
- Integration with the program NAMD, a fast, parallel, and scalable molecular dynamics program developed in conjunction with VMD in the Theoretical and Computational Biophysics Group at the University of Illinois. See the NAMD WWW home page for more info: `http://www.ks.uiuc.edu/Research/namd/` VMD works well with projected display systems like the 3-D Projection Facility maintained by the Theoretical and Computational Biophysics Group.
- VMD can be used to concurrently display and interact with a running NAMD simulation
- Using VMD as a Web/MIME Helper Application
- VMD script library
- Portable and Intuitive GUI
- Interfacing with MD Programs (i.e. SIGMA, NAMD)
- Interactive MD and VR Device Support
- Making Movies with VMD

6.2.5 PyMOL

PyMOL is a user-sponsored molecular visualization system on an open-source foundation. After years of development and testing in the open-source community, PyMOL has established itself as a leading software package for customization of 3-D biomolecular images, with more than 600 settings and 20 representations to provide users with precise and powerful control.

PyMOL can interpret over 30 different file formats from PDB files to multi-SDF files to volumetric electron density maps. PyMOL's straightforward graphical user interface allows first-time and expert users alike to create stunning 3-D images from their favorite file formats. Images and movies can then be saved in a cross-platform Session file, ensuring that every object position, atom color, molecule

representation, molecular state, frame, and movie can be viewed by colleagues exactly as intended. PyMOL runs on Windows, Linux and OSX

6.2.6 Chemcraft

Chemcraft is a graphical program for working with quantum chemistry computations. It is a convenient tool for visualization of computed results and preparing new jobs for the calculation. Chemcraft is mainly developed as a graphical user interface for Gamess (US version and the PCGamess) and Gaussian program packages. For working with other formats of calculations, the possibility to import/export coordinates of atoms in text format can be easily used. Chemcraft does not perform its own calculations, but can significantly facilitate the use of widespread quantum chemistry packages. Chemcraft works under Windows and Linux (but the Linux version has some disadvantages). The main capabilities of the program include

- Rendering 3-dimensional pictures of molecules by atomic coordinates with the possibility to examine or modify any geometrical parameter in the molecule (distance, angle);
- Visualization of Gamess, Gaussian, NWChem, ADF, Molpro, Dalton, Jaguar, Orca, QChem output files: representation of individual geometries from the file (optimized structure, geometry at each optimization step, etc), animation of vibrational modes, graphical representation of gradient (forces on nucleus), visualization of molecular orbitals in the form of isosurfaces or colored planes, visualization of vibrational or electronic (e.g. TDDFT) spectra, possibility to show SCF convergence graph, and some other features;
- Different tools for constructing molecules and modifying molecular geometry: using standard molecular fragments, "dragging" atoms or fragments on the molecule's image, utility for setting a point group, and other possibilities;
- Producing publication-ready images of molecules in customizable display modes, containing required designations (labels, lines, etc);
- Some additional utilities for preparing input files: visual construction of Z- matrixes, automatic generation of input files with non-standard basis sets, converting MOs read from an output file into the format of input file.

The program combines advanced graphical user interface and wholesome features designed for practical use. Chemcraft provides very detailed structured visualization of output files, based on dividing a file into separate elements and presenting them in hierarchical multi-level list; this feature allows one to easily analyze complicated files, such as scan jobs, IRC jobs, or multi-job calculations. The graphical engine of Chemcraft does not require any hardware acceleration.

Chemcraft is a commercial software. The cost of the Windows version is currently 105 USD for academic users and 190 USD for others. Freeware version of Chemcraft for Windows with several limitations and nag- screens is also available for download.

6.3 Conclusion

The increasing appreciation of the inter-relatedness and multifactoriality of pre-clinical drug discovery makes data analysis and visualization a burgeoning and active field spanning through biosciences , mathematics to visual psychology. The efforts has greatly helped to reduce attrition rates of drug candidates during the many years that are usually involved in drug development.

The number of years usually involved in pre-clinical drug development have also been reduced

THANK YOU FOR THE APT ATTENTION

MATLAB/SCILAB Training Course, Vol. 1, pp. 82–91, 2011

Received 15 November 2011

CHAPTER 7

DATA ANALYSIS/ VISUALIZATION IN SOCIAL SCIENCES: CURRENT CHALLENGES AND FUTURE TRENDS

by

EZICHI A. ITUMA* & EMMA EZEANI+

***Coordinator, Combined Social Sciences**

+Dean, Faculty of the Social Sciences

University of Nigeria, Nsukka

7.1 Introduction

This paper examines the challenges of data analysis in the disciplines of social sciences. Since the concern of social sciences is that of order in the society as it relates to human beings in their social interaction it becomes necessary that human activities and responses to social problems be given primary attention. Incidentally human behaviour is sometimes determined by goals that are hidden in the mind characterised by idiosyncratic impulse. As a scientific research data analysis must follow a systematic and observable order. Interestingly scholars have noted that understanding the natural world can be termed singularly hermeneutic since the natural world cannot interpret back but there is a dialogical relationship between researchers and their human subject matter which is doubly hermeneutic, because human beings can and do interpret back, which creates an interpretive loop.1 As a result, social research requires methodologies that address the complexities of the subject matter.

7.2 Social Sciences Methodologies

It has earlier been noted that the major concern in the Social Sciences is the problem of order in the society. Human beings in a series of integrated relationships make up the society. One way that social order has been theorized is according to the degree of integration of cultural and social factors. It deals with habitual and behavioural responses to social order. It is has been noted that "social research reflects society, and society itself is diverse, multifaceted, and composed of many antagonistic groups. It follows that the goals of social research are multiple and sometimes contradictory. Today, no single goal dominates social research."2 In Social Sciences, as a result, one identifies five methodological models in qualitative research which includes ethnography, grounded theory, hermeneutics, empirical phenomenological research and heuristic research. Ethnography is a social scientific methodology very popular in anthropology and related disciplines. It studies "people, ethnic groups and other ethnic formations, their ethnogenesis, composition, resettlement, social welfare characteristics, as well as their material and spiritual culture"3 Grounded theory is another social scientific methodology that "was originally developed by Glaser and Strauss in the 1960s. The self-defined purpose of grounded theory is to develop theory about phenomena of interest. But this is not just abstract theorizing they're talking about. Instead the *theory* needs to be *grounded* or rooted in observation -- hence the term."4 Empirical phenomenology engages in a "structural analysis of the participants' accounts of a phenomenon in order to determine the essential components of that experience ... in the sense that the end result of the analytical process is a description of the structure of the experience provided by the participants."5 Hermeneutics is the study of interpretation theory, and can be either the art of interpretation, or the theory and practice of interpretation.6 This method is found in traditional hermeneutics, profusely used in biblical studies and modern hermeneutics, used in philosophy. Heuristic research is a method of self-search and self discovery used mostly in psychology. It is a "way of inquiring into what Wilber has identified as the Upper Left quadrant, or individual internal experience, and what Polanyi referred to as tacit knowledge, which is deeply embedded knowledge not normally available to conscious awareness."7 In other words it is "a process of internal search through which one discovers the nature and meaning of experience and develops methods and procedures for further investigation and analysis."8 We should note the overlapping tendency in the methods. This is the string that ties them within the social sciences domain. Please note that the way we approach phenomenological research in Social Sciences is not always the way it could be approached in Physical Sciences. Yet the connecting string is that of observable phenomena. The essence of these presentations is to disassociate ourselves from the erroneous conclusion that in the Social Sciences there is a consensus of methodology that must be highlighted in dominance against all other ones. Though in the above methodologies there are computer-mediated softwares that have been developed to make research easy and fast, it is however in the quantitative research that we find so much of computer-mediated softwares. There is no doubt that electronic processing and presentation of information captures the interest of the end users of research findings more than the old traditional ways. The challenge is that people like to have well analysed data with effective visualization but only a very small group, and sometimes insignificant few, will want to represent research findings with such interest-capturing

techniques.

7.3 Data Analysis

This is a process of inspecting, cleaning, transforming, and modelling data with the goal of highlighting useful information, suggesting conclusions, and supporting decision-making.9 It is a scientific process that enables one to make better decisions in a system, industry, business organisation or to verify or disprove existing models or theories. Different approaches and techniques are adopted depending on the nature of the research and discipline a researcher belongs to. It is generally divided into Exploratory Data Analysis which emphasises discovery, Confirmatory Data Analysis which sustains or disproves existing hypothesis or theory, and Qualitative Data Analysis, used to examine non-numerical data like words, photographs, maps, videos etc. Qualitative Data Analysis is an approach which is very popular in the Social Sciences and has, as a result, received more attention than the rest. The first two analytical approaches could be summarized in Quantitative Data Analysis.

7.3.1 Note on differences

Though we try as much as possible to examine these different ways of data analysis it is necessary to note that one form of analysis is always loosely present in the other form of data analysis. We do not, however, pretend that there are not different types of qualitative research, the traditional and the critical modes, and their different special emphasis of methods of data collection, analysis and interpretation. We have therefore limited ourselves to what is presented here, guided by the topic assigned.

7.3.2 Qualitative Data Analysis

In Social Sciences the aim of Qualitative analysis is "to gather an in-depth understanding of human behaviour and the reasons that govern such behavior. The qualitative method investigates the why and how of decision making, not just what, where, when."10 It is a research that allows the data to speak for itself, in line with some theoretical framework. The reason for this is because in qualitative data analysis it is very difficult for the neutrality and unbiased personality of the researcher to be established. Neutrality and unbiased position is highly required in qualitative analysis, however. In this form of data analysis diversity in responses as well as the ability to adapt to new developments feature very prominently. Theoretical Framework is a theoretical approach that guides data analysis. In theoretical framework a number of theories are adopted, depending on the intention and nature of the topic or problem being solved. Research problems are viewed and variables interpreted from the vantage point of theoretical framework.

7.3.3 Quantitative Data Analysis

Quantitative Data Analysis is used in the Social Sciences mainly in the following disciplines, Economics, Psychology and Geography. Even in these disciplines it is not as regularly used as that of qualitative research. We do not, however, imply that other Departments/ disciplines in Social Sciences are not involved in Quantitative research/ analysis. We rather imply popularity of usage. In this research the software that is popularly used is the Statistical Package for Social Sciences, SPSS. This is a package that is convenient for large amount of numerical data.

7.3.4 A word for SPSS

Statistical Package for Social Sciences, SPSS, is a computer programme used for numerical data analysis, data management and data documentation. It was acquired from the Predictive Analytics SoftWare (PASW) by IBM in 2009.

Statistics included in the base software:

- Descriptive statistics: Cross tabulation, Frequencies, Descriptives, Explore, Descriptive Ratio Statistics
- Bivariate statistics: Means, t-test, ANOVA, Correlation (bivariate, partial, distances), Nonparametric tests
- Prediction for numerical outcomes: Linear regression
- Prediction for identifying groups: Factor analysis, cluster analysis (two-step, K-means, hierarchical), Discriminant.11

When cases are studied in SPSS the first thing to identify are the variables. Values are assigned to the variables. The strenuous aspect of the package, however, is the keying of the data into the software. Another challenging aspect is the identification the right kind of variables.

Not only does SPSS require that you select a sufficient number of variables to produce output, it also requires that you choose the right kinds of variables. If a categorical variable is required for a certain slot, SPSS will not allow you to choose any other kind. Whether the output makes sense is up to you and your data, but SPSS makes certain that the choices you make can be used to produce some kind of result.12

7.4 Visualization

Data visualization is the visual representation of data in charts, maps and graphics so as to achieve the effective communication goal. In this effort it is "used to describe any technology that lets corporate executives and other end users "see" data in order to help them better understand the information and put it in a business context."13 The focus is to provide "a graphical interpretation of a company's data so that it can be analyzed from different perspectives."14 The researcher should always bear the end users in mind while representing the data analysis. A research that is not packaged in a way that lets the users understand what is required is a research that is still in the mind of the researcher. The society wants solution to problems. Clarity of thought is a very important feature of research because the mind must be read by those who are outside. It is possible for the owner of a house to locate his belongings even when the room dark. If some other persons are required to locate positions of items then the room needs illumination. If a research provides possible solutions to social problems, yet those who are supposed to implement the findings so as to create favorable societal atmosphere hardly understand the findings, the researcher has failed to communicate. This problem is serious even in the University, among academics. Sometimes we supply data without having a good understanding what the data represents. Demonstrate the power of positive visualization we may use the Senate paper on examination results. Departments supplied the data and the analysis was made by the Exams Unit of the Registrar Department. However, when we visualized the data analysis, almost all the Departments protested that we indicted them wrongly.

7.5 Data Analysis/Visualization of Senate Paper on approval of Examination results

Table 7.2: Distribution of 2010/2011 Graduating Students' Results in the Faculty of Agriculture

Departments	Grad	Total	% Processed	Failure	Total Unprocessed	% Unprocessed
Agric Economics	41	21	51.22	9	20	48.78
Agric Extension	19	17	89.47	2	2	10.53
Animal Science	28	8	28.57	2	20	71.43
Crop Science	10	5	50	5	5	50
Food Sc. & Technology	58	23	39.66	4	35	60.34
Home Sc Nutrition & Diet	45	16	35.56	15	29	64.44
Soil Science	4	4	100	-	-	0
Total	205	94	45.85	37	111	54.15

Table 7.4: Distribution of 2010/2011 Graduating Students' Results in the Faculty of the Social Sciences

Departments	Grad	Total	% Processed	Failure	Total Unprocessed	% Unprocessed
Economics	230	116	50.43	54	114	49.57
Geography	42	28	66.67	9	14	33.33
Philosophy	39	25	64.10	6	14	35.90
Political Science	190	69	36.32	20	55	63.68
PALG	229	130	56.77	44	99	43.23
Psychology	160	73	45.63	27	87	54.37
Religion	16	16	100	-	-	0
Social Work	70	42	60	15	28	40
Soc & Anth	80	59	73.75	17	21	26.25
Total	1056	558	52.84	192	498	47.16

The different Departments may not fully understand the meaning of the data they had generated for the Senate meeting until it is represented in this form. Some Departments and Faculties may even argue that they did not supply the data.

7.5.1 Reality of the meaning of the data base on visualization

Failure grades in the results

1. Some lecturers released results so late that the students had no opportunity to register them again, eg, 200 level result released when the students are in final year.
2. The students did not understand the lecture, etc
3. The lecturer either did not teach or does not know how to teach or many students could not be made even after the four years class lectures, assignments, industrial training, etc.

Other Reasons for unprocessed results

1. Results were released in piecemeal such that single results outnumbered the group results. As a result of this, such single result sheets were either lost on transit or buried inside some mega sheets. Even the dispatch staff may have considered this single result of less importance such that he forgot to dispatch.

2. Exams Unit of the Registrar had very poor filing habit such that results sheets got "missing",
3. Processing of files were not properly monitored. So, many files are either left in Records or in Admissions Office
4. etc

7.5.2 Challenges

It is not arguable that teaching and learning become interesting, interactive and less boring when we adopt effective visualization techniques. Visualization is very necessary if we must teach and express our research findings with good intentions to impact on our audience. There are many challenges and problems militating against effective data analysis and visualization among the staff and students of social sciences in quantitative research and qualitative research.

1. **Lack of patience** Both staff and students are in a hurry to conclude research works. As a result of this, coding and keying data into the systems become a nightmare.
2. **Unavailability of software:** With the high level of software piracy in Nigeria, it is sometimes impossible to find original softwares. Many pirated softwares are hardly properly installed. As a result of this such softwares cannot function properly. It can be very disgusting that large amounts of data were patiently keyed into the system only for the software to fail. Not only that the original softwares are very costly, they are not even available, most times in the market.
3. **Lack of interest** It is a shock sometimes that in a graduating class of three hundred students only five of the students carried out quantitative research, effectively analysing data with computer softwasre. Among staff and students the interest is hardly present. If one is in doubt I will suggest a look at the journals in the Faculty to establish the fact that members of the academia are no longer interested in this. A look at the research projects in the Departments will also establish the lack of interest among the students. Hitachi Star boards were installed in the Faculties and the lecturers rather preferred to use white board makers on them. In some of the Faculties the Deans decided not even to open the carton that contains the softwares. Even the Nigerian Universities Management Information System (NUMIS) exam computation software is hardly used in the departments.
4. **High level of "Grade syndrome"** High level of "Grade syndrome" among students is a big problem. Students attend classes where they are taught how to do research but they want to pass exams and will learn the methods. They want to have good grades at little stress. Few occasions we hear students describe a lecturer as one who stresses students. If such courses have alternative electives then no student will want to go to the stressful areas.

5. **Computing Systemsnot available** Computing Systems are not available for students to put into practice what they learn. The only option will be to look for ventures who display billboard: "Come and run your analysis without stress," "Come and have readymade Term Papers and Projects at very little cost." The moment a student realizes that he could pay a very little token to run analysis or getting a research Project by merely paying a little token of money he will not be bothered about learning the techniques of data analysis or rudiments of research.

7.5.3 Suggestions

1. The University system is a research institution and should therefore provide enough computer systems in the departments for the students.

2. Both staff and students should be encouraged to use computer-mediated softwares in their research, perhaps by appointing facilitators in every department.

3. Visual display equipment should be installed in every Department and experts trained to man the equipment, retrain staff and monitor the use of such equipment. The Hitachy Star Board was never used in the University yet it has been installed in the Faculties. The Nigerian Universities Management Information System exam computation software (NUMIS) is another example.

4. ICT compliance should be given priority consideration in staff appraisal for promotion. Note that in webometric ranking of Universities in the world visibility in the internet is the first port of call.

7.5.4 Trends

Many students spend so much time in the internet but most times it is in facebook and other chatting devises. They watch and download films and hardly aware that they could have unlimited access to research materials in the internet. When this is the case we end up producing graduates who go into the labour market unable to solve challenging problems which would be easily solved if the computer-mediated softwares were handy. Modern approach to problems is no more the old traditional approach. Precision is the keyword now and it is automated. Yes, virtually everything is automated to produce results. Without producing graduates who automate their industrial input in the labour market we fail in our duty.

7.6 Conclusion

It is completely out of place to conclude that research and learning are getting on effectively without effective data analysis and visualization. There is no field of research that does not require data

analysis and visualization. Systematic inspecting, cleaning, transforming, and modelling of data with the goal of highlighting useful information must consider modern techniques and recent software developments. The tertiary institution distinguishes itself in research. Modern trend in research makes so much use of computer-mediated softwares and we fail if we want to go back to the old techniques. A good scholar is a scholar who understands the times and makes use of the available tools to produce good results.

References

1. **See D. Hiley, J. Bohman, & R. Shusterman, (Eds)**, The interpretive turn: Philosophy, Science, culture (pp. 17-24) Cornell University Press, London, in T. Kuhn, The Natural and the Human Sciences in Richard S. Zayed, The Changing Nature of the Phenomenological Method: Lessons Learned from Dialogical Psychotherapy Research www.janushead.org/10-2/Zayed.pd, for Bohman, Hiley and Shusterman, 1994, p. 4

2. **The Goals of Social Research,** http://poli.haifa.ac.il/˜levi/res/mgsr1.htm, Retrieved May 5, 2011

3. **http://en.wikipedia.org/wiki/Ethnography#cite_note-1,** Retrieved May 5, 2011

4. **http://www.socialresearchmethods.net/kb/qualapp.php**, Retrieved May 6, 2011

5. **Katie-Ann Berrya; Kent C. Kowalskia; Leah J. Fergusona; Tara-Leigh F. McHug**, An empirical phenomenology of young adult women exercisers' body selfcompassion (London: Routledge Informa Ltd, 2010) http://dx.doi.org/10.1080/19398441.2010.517035, Retrieved May 7, 2011

6. **"Hermeneutics"** http://en.wikipedia.org/wiki/Hermeneutics, Retrieved May 8, 2011

7. **Sandy Sela-Smith** "Heuristic Research: A Review and Critique of Moustakas's Method" Journal of Humanistic Psychology, 2011, http://jhp.sagepub.com/content/42/3/53.abstract, Retrieved May 9, 2011

8. Heuristic Research Design, Methodology and Application, http://www.cosmicplay.net/Method/Extra/extraheur1.html, Retrieved May 10, 2011

9. http://en.wikipedia.org/wiki/Data_analysis, Retrieved May 10, 2011

10. http://en.wikipedia.org/wiki/Qualitative_research, Retrieved May 10, 2011

11. **SPSS**, http://en.wikipedia.org/wiki/SPSS, Retrieved May 10, 2011

12. **Arthur Griffith**, How SPSS (Statistical Package for the Social Sciences) Works, http://www.dummies.com/how-to/content/how-spss-statistical-package-for-the-social-scienc.html, Retrieved May 10, 2011

13. http://searchbusinessanalytics.techtarget.com/definition/data-visualization, Retrieved May 11, 2011

14. **Visualization,** http://businessintelligence.ittoolbox.com/directory/other/data-vis, Retrieved May 11, 2011

Table 7.1: Faculty of Social Sciences Unprocessed Results Chart

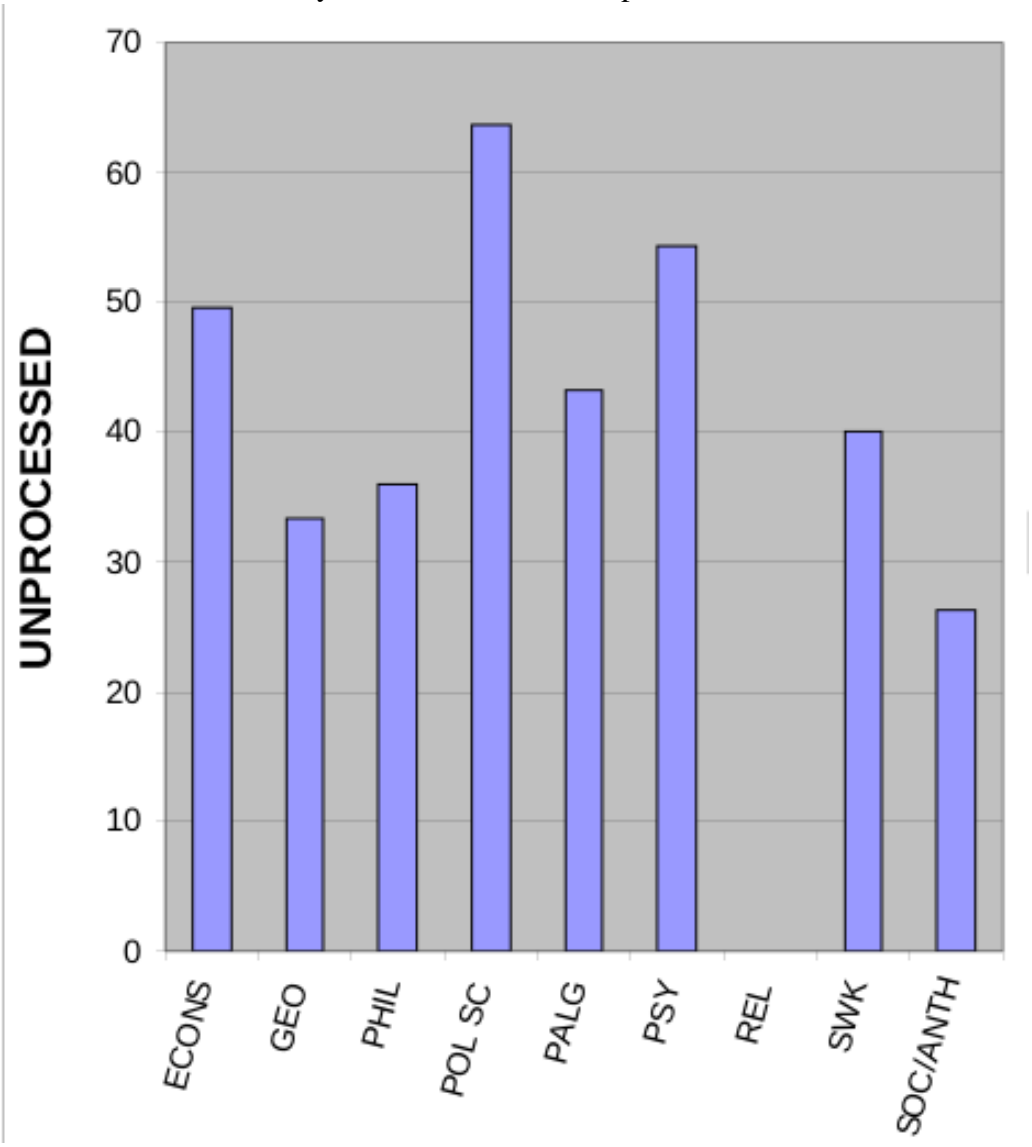


Table 7.3: Faculty of Social Sciences Unprocessed Results Chart

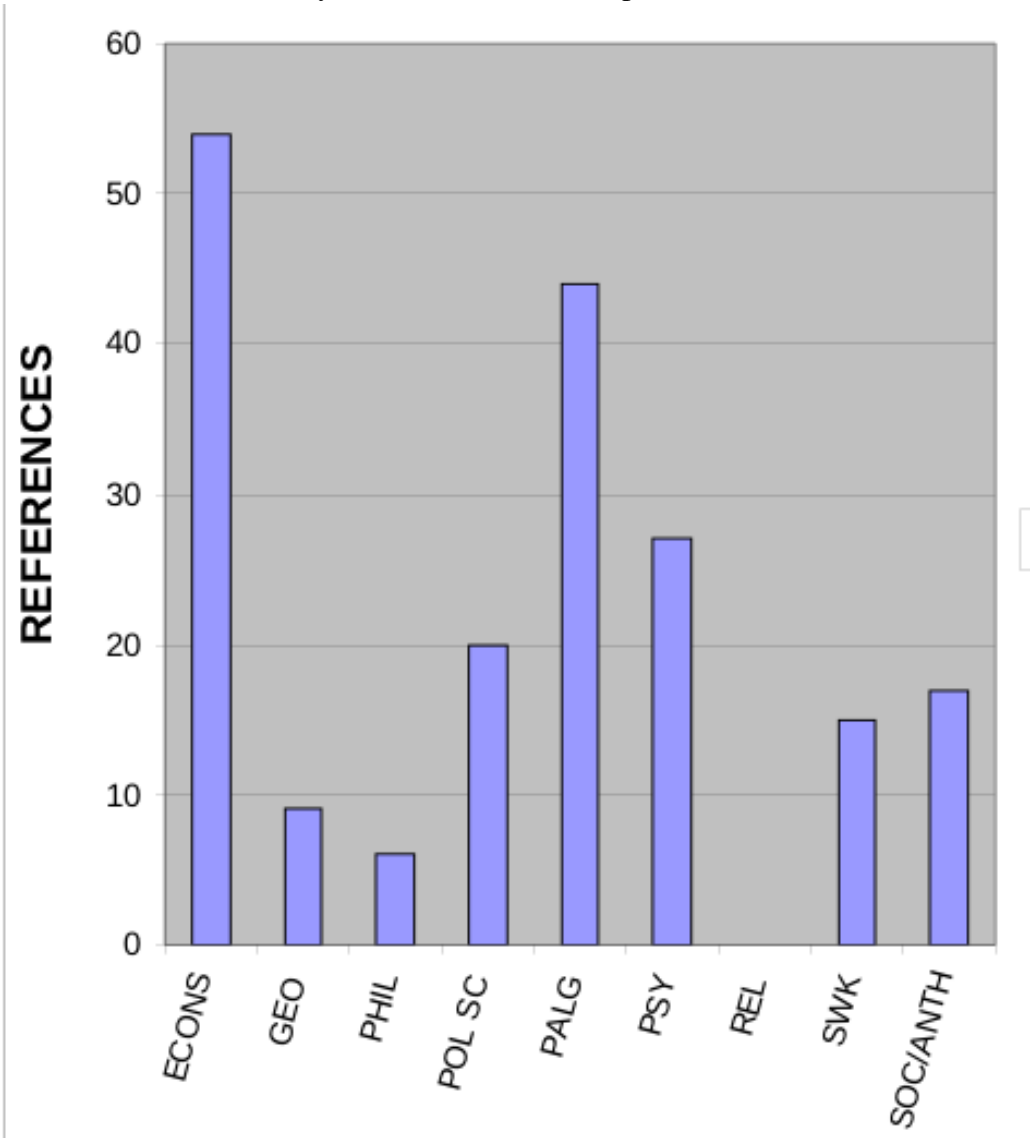


Table 7.5: Faculty of Agriculture Failure Chart

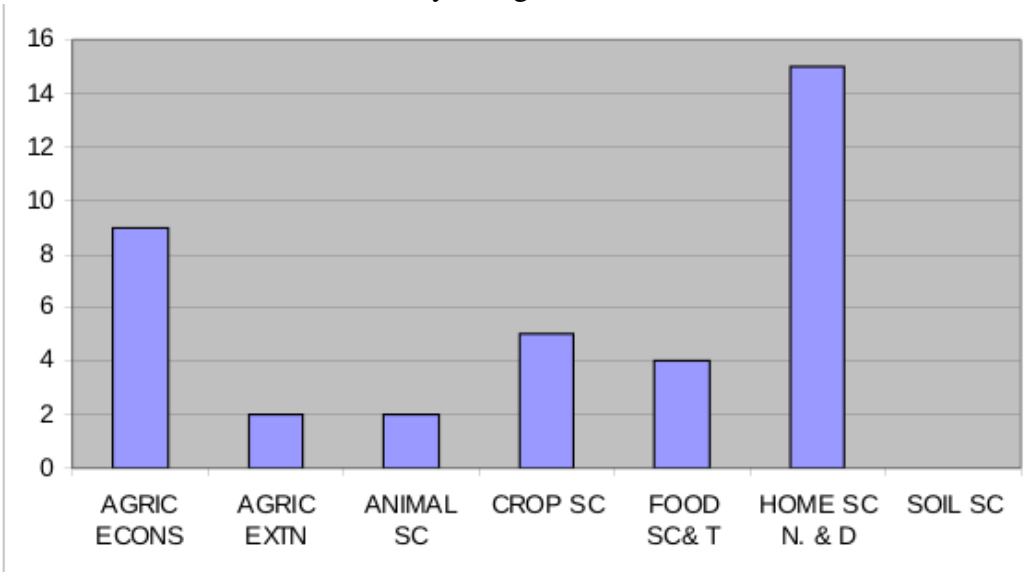
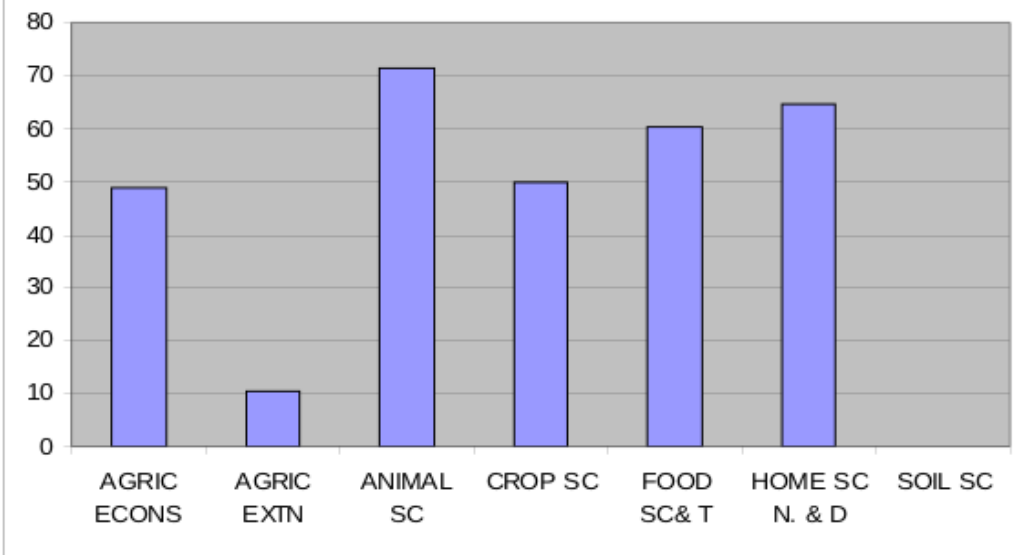


Table 7.6: Faculty of Agriculture Unprocessed Results Chart



MATLAB/SCILAB Training Course, Vol. 1, pp. 95–133, 2011

Received 20 November 2011

CHAPTER 8

DATA/SYSTEMS ANALYSIS WITH MATLAB/SCILAB

by

S. O. ENIBE
Department of Mechanical Engineering
University of Nigeria, Nsukka

8.1 Introduction

Data analysis is an important step in any research or development project. This will usually involve the conversion of the raw project output data into forms that are more suited for interpretation or presentation to the final users of the data. In this document, a review of the most common project output data types will be undertaken, followed by a suggested procedure for data analysis. The document will then be concluded with typical examples from recent projects.

Several computer packages have been developed to facilitate efficient data analysis and visualization. These range from very high level structured programming languages such as FORTRAN, C or C++, to the interpreted languages such as BASIC, MATLAB, SCILAB, FREEMATH, PYTHON, etc and on to special purpose dedicated software such as Statistical Package for Social Sciences (SPSS), MINITAB, among others.

Each of these packages have their merits/demerits, but these will not be further considered in this work. It is worthy to note, however that in recent years, MATLAB/SCILAB have gained prominent positions among practicing engineers, scientists and students as very reliable and easy to use tools for data analysis, visualization, modeling and simulation.

MATLAB and SCILAB are high level computer languages built around the idea of an interactive programming environment. The great advantage of an interactive system is that programmes can be tested and debugged quickly, allowing the user to concentrate more on the principle behind the programme and less on programming itself. In this way, practical problems can be solved in much shorter time than would be possible with FORTRAN, C, C++, C# or even BASIC. This is because both MATLAB and SCILAB contain large numbers of functions that utilize proven numerical libraries.

As a result, common tasks (such as solution of simultaneous equations) can be accomplished with a single function call! Further, there is extensive graphics support that allow the results of computations to be plotted with a few statements.

MATLAB and SCILAB are similar in many respects, and employ largely the same syntax. However, MATLAB is a commercial software that requires the payment of a fee before legal use, while SCILAB is freely available from the developers' website (*http://www.scilab.org/download*).

8.2 Applications

Applications include: heterogeneous simulations and data-intensive analysis of very complex systems and signals, comprehensive matrix and arrays manipulations in numerical analysis, finding roots of polynomials, two- and three-dimensional plotting and graphics for different coordinate systems, integration and differentiation, signal processing, control, identification, symbolic calculus, optimization, etc. The goal of MATLAB/SCILAB is to enable the users to solve a wide spectrum of analytical and numerical problems using matrix-based methods, attain excellent interfacing and interactive capabilities, compile with high-level programming languages, ensure robustness in data-intensive analysis and heterogeneous simulations, provide easy access to and straightforward implementation of state-of-the-art numerical algorithms, guarantee powerful graphical features, etc. Due to high flexibility and versatility, the MATLAB/SCILAB environments have been significantly enhanced and developed during recent years. This provides users with advanced cutting-edge algorithms, enormous data-handling abilities, and powerful programming tools.

8.2.1 Basic functions

In order to make the implementation of the vast capabilities of SCILAB easy to use for the average user, all applications have been divided into a very large number of functions that can generally interact in a seamless manner. These functions are then arranged into some 60 categories, each category representing a class of applications. The complete list of categories in any installation can be obtained by typing the command **help** in the command line. The list of unctions available under each category can be obtained by expanding the category folder in the help browser.

8.2.2 Toolboxes

In addition to the basic functions discussed above, both SCILAB and MATLAB have developed large numbers of toolboxes or additional collections of functions to handle specilized applications. For example, there are aabout 18 different toolboxes avaialbe in SCILAB for statistical and data analysis alone. Altogether, at least 110 toolboxes are available, with more being created according to user needs. A list of available toolboxes in SCILAB is shown in table 8.1.

Table 8.1: Available toolboxes in SCILAB

accsum	Accurate summation algorithms
ampl_toolbox	An interface to load .nl files created by AMPL
ANN_Toolbox	ANN Toolbox
apifun	Check input arguments in macros.
arfit	Multivariate Autoregressive Model Fitting
assert	A collection of predicate (i.e., trueâfalse) functions
celestlab	CNES Space mechanics toolbox for mission analysis
CLUSTER	This toolbox implements functions for clustering and for evaluating clustering algorithms.
condnb	Evaluates the condition number of functions.
convertlatin	functions to convert to UTF-8 to latin (and reverse)
csv_readwrite	Fast functions to read and write csv files
CUTEr	Testing environment for optimization and linear algebra solvers
dace_scilab	This is a conversion of the well known DACE kriging toolbox for Matlab.
dbldbl	Double-Double floating point numbers
dde_toolbox	Dynamic Data Exchange client for Scilab
DeCaToKi	This work present a serie of functions to analize state-space and LPV descriptor systems
diffcode	Automatic differentiation
digimetrie_Tools	Provide functions ans Xcos Blocks for control of Digimetrie's boards
dispmat	Display matrices graphically
edflib	load/write edf/bdf-files
emd_toolbox	Toolbox for Empirical Mode Decomposition of 1-D, 2-D and more dimesional signals.
EMTTOOL	An electromagnetics toolbox for scilab
femtruss	A Truss finite element code for scilab
Financial	Module is about risk measure and management, asset allocation, and pricing
floatingpoint	Functions to manage floating point numbers

fmincon	Nonlinearily constrained multivariable optimization solver
fsmtbx	Fast factorization of matrices with displacement structure
Global_Optim_toolbox	Global optimization toolbox for solving unconstrained polynomial optimization problems
grocer	Comprehensive econometric toolbox
guibuilder	A Graphic User Interface Builder
guimaker	A toolbox for creating GUIs with a minimum of programming.
helptbx	Update the help of a module automatically.
huffcomp	This small toolbox shows the principles of huffman coding.
hyb_auto_module	Toolbox enables visual modeling of hybrid automata in Xcos
HYDROGRv50	Models and function for operational hydrology
ica_toolbox	Toolbox containing algorithms for independent component analysis and blind source seperation
identification	Tools for identification of dynamical systems
imsls	Iterative Methods for Sparse Linear Equations
indinf	Algorithms to calculate influences of graphs
intprbs	Provides 23 functions for multidimensional integration.
IPD	This toolbox implements functions for object detection.
ipoptlib	Binary versions of the Ipopt library
JIMS	Module to manipulate Java objects from Scilab
json	A Simple JSON Toolbox for Scilab
krisp	Kriging based regression and optimization package for Scilab.
lcc_windows	LCC-win32 support for Scilab
linalg	A collection of algorithms for linear algebra
lolimot	A fast neural network & LOcal LInear MOdel Tree
lowdisc	Provides Halton, Sobol and other sequences.
lsitbx	Scilab 5 toolbox for optimal stable inversion of linear time-invariant systems.
makematrix	A collection of test matrices.
Mathieu	Solve Mathieu equations
MatrixMarket	Read and Write Matrix Market formatted files
Max_Plus_Algebra	Collection of some theory and applications in Max Plus Algebra
MDPtoolbox	Markov Decision Processes toolbox
metanet	Graph and Network toolbox
microwave	RF and Microwave Tools for Data Analysis
mingw	Dynamic link with MinGW for Scilab on Windows
minphase_toolbox	Design a stable discrete IIR filter with arbitrary gain or phase.
mkltools	tools for optimized Intel Math Kernel Library used by Scilab

module_lycee	Scilab pour les lycÃ©es
montesci	Monte Carlo simulator with GUI
MPScilab	Arbitrary precision mathematical computing
nan	A statistics and machine learning toolbox
NIDAQ	NI-DAQmx interface with Scilab
NISP	Non Intrusive Spectral Projection
nistdataset	NIST Statistical Reference Datasets
NTG	This module is dedicated to network topologies analysis.
number	Integers algorithms
opc_client	OLE for Process Control Toolbox for Scilab
optkelley	Scilab software for Iterative Methods for Optimization
plotlib	"Matlab-like" Plotting library for Scilab
PSO	The PSO on Scilab
quapro	Linear and Linear Quadratic Programming
regtools	A toolbox for linear and non linear regression analysis
RLDesign	Simple GUI for Control Design Using Root Locus
rltool	A GUI toolbox for designing SISO systems
scetoexe	example to embed a .sce in a windows .exe
sci_ipopt	A Non-linear Interior Point Optimization Solver
scibench	A collection of benchmarks
scicobyla	An interface to the COBYLA optimization method.
scidb	Connect to databases, execute queries, load data to/from DB from/to Scilab environment
scidemo	A collection of demonstrations
sciGPGPU	Gpu Computing for Scilab
scilab2c	Translate Scilab code into C code
Scilab_XLL	Scilab Xll Add in for Excel (97 to 2007)
scilib_geo	A collection of functions for use in geodaesie and astronomie
sciopustc	Opus Test Case Module
scipad	Scipad 7.20
serial	A toolbox for communication over a Serial Port in Scilab
simplex	This package contains the simplex optimization method
siseli	Serial ports communication on Windows (Scilab 5.3+)
SIVP	SIVP intends to do image processing and video processing tasks
SMP	Some built-in to manage Scilab process on Windows
sndfile_toolbox	Read & write sound files
socket_toolbox	Basic socket functions for Scilab

specfun	A collection of special functions
spilu	Incomplete LU factorizations
splspc	Sparse Least Squares Pre-Conditioned methods
stftb	Toolbox developed for the analysis of non-stationary signals using time-frequency distributions.
stixbox	Statistics toolbox
sudoku	Solve sudoku puzzles
swt	mimic matlab wavelet toolbox
uncprb	Provide 35 unconstrained optimization problems
Wavelab	WaveLab is a library for wavelet analysis, wavelet- packet analysis, cosine-packet
WriteXmlExcel	Write Excel xml files
xcos_toolbox_skeleton	Skeleton to create an Xcos palette (implementation of blocks) toolbox
xls_link	xls_link (Automation link with Excel) A easy way to call excel(TM) from Scilab
xmllab	XMLlab is a simulation authoring environment using XML and Scilab
xmltodocbook	converts scilab 4.x help files to scilab 5 help files

8.3 Project Data Types

A typical research, development or field project will normally result in some data which could be a number, string or signal. Some of the data may be generated by direct measurement or observation or from computer simulation. Each of these will now be considered briefly, while further details may be found in Baudin (2010), Baudin *et al* (2010), Lyshevski (2003), among others.

8.3.1 Numerical Data

The numerical data may be of type integer, real number, or complex number. Data in this format can be manipulated in a very straightforward manner by making use of built-in functions for manipulating matrices. Full details are presented in the software documentation, user manual and other literature (see for example Baudin (2010), Campbell *et al* (2006) and Kiusalaas' (2005).

8.3.2 String Data

Some projects generate string objects as one of the major outputs. These could range from single-character text objects such as alphabets (eg A to Z, a to z) digits 0 to 9 and punctuation marks; to longer

strings made up of a mixture of characters. These may be used to describe list structures, names, user responses and the like. Data structures of this nature are common in field projects involving the use of structured or unstructured questionnaire, discussions, and the like.

8.3.3 Signals/Stream Data

In experimental projects, the output from a measuring device may be a signal or stream data from an instrument. There are extensive built-in functions for analyzing signals and stream data in general. A one-line desciprion of the 71 signal processing functions in SCILAB are shown in table 8.2. Comparable functions are also available in MATLAB.

Table 8.2: Signal processing functions in SCILAB

Function	Description
analpf	create analog low-pass filter
bilt	bilinear or biquadratic transform SISO system given by a zero/poles representation
buttmag	Power transmission of a Butterworth filter
casc	cascade realization of filter from coefficients
cepstrum	cepstrum calculation
cheb1mag	response of Chebyshev type 1 filter
cheb2mag	response of type 2 Chebyshev filter
chepol	Chebychev polynomial
convol	convolution
corr	correlation, covariance
cspect	two sided cross-spectral estimate between 2 discrete time signals using the correlation method
czt	chirp z-transform algorithm
detrend	remove constant, linear or piecewise linear trend from a vector
dft	discrete Fourier transform
ell1mag	magnitude of elliptic filter
eqfir	minimax approximation of FIR filter
eqiir	Design of iir filters
faurre	filter computation by simple Faurre algorithm
ffilt	coefficients of FIR low-pass
fft	fast Fourier transform.
fft2	two-dimension fast Fourier transform

Table 8.2 continued: Signal processing functions in SCILAB

Function	Description
fftshift	rearranges the fft output, moving the zero frequency to the center of the spectrum
filt_sinc	samples of sinc function
filter	filters a data sequence using a digital filter
find_freq	parameter compatibility for elliptic filter design
findm	for elliptic filter design
frfit	frequency response fit
frmag	magnitude of FIR and IIR filters
fsfirlin	design of FIR, linear phase filters, frequency sampling technique
group	group delay for digital filter
hank	covariance to hankel matrix
hilb	FIR approximation to a Hilbert transform filter
hilbert	Discrete-time analytic signal computation of a real signal using Hilbert transform
iir	iir digital filter
iirgroup	group delay Lp IIR filter optimization
iirlp	Lp IIR filter optimization
intdec	Changes sampling rate of a signal
jmat	row or column block permutation
kalm	Kalman update
lattn	recursive solution of normal equations
lattp	lattp
lev	Yule-Walker equations (Levinson's algorithm)
levin	Toeplitz system solver by Levinson algorithm (multidimensional)
lindquist	Lindquist's algorithm
mese	maximum entropy spectral estimation
mfft	multi-dimensional fft
mrfit	frequency response fit
percentasn	elliptic integral
percentk	Jacobi's complete elliptic integral
percentsn	Jacobi 's elliptic function
phc	Markovian representation
pspect	two sided cross-spectral estimate between 2 discrete time signals using the Welch's average periodogram method.

Table 8.2 continued: Signal processing functions in SCILAB

Function	Description
remez	Remez exchange algorithm for the weighted chebyshev approximation of a continuous function with a sum of cosines.
remezb	Minimax approximation of magnitude response
rpem	Recursive Prediction-Error Minimization estimation
sincd	digital sinc function or Direchlet kernel
srfaur	square-root algorithm
srkf	square root Kalman filter
sskf	steady-state Kalman filter
syredi	Design of iir filters, syredi code interface
system	observation update
trans	low-pass to other filter transform
wfir	linear-phase FIR filters
wiener	Wiener estimate
wigner	'time-frequency' wigner spectrum
window	compute symmetric window of various type
yulewalk	least-square filter design
zpbutt	Butterworth analog filter
zpch1	Chebyshev analog filter
zpch2	Chebyshev analog filter
zpell	lowpass elliptic filter

8.3.4 Simulated Data

Simulation and optimization is becoming increasingly important in all areas of research and development. As a result, many projects result in significant amounts of simulated data which may require further processing. Such data would usually be in any of the three formats already presented, and could thus be handled in the same way.

8.4 Graphics

MATLAB and SCILAB have excellent support for the generation of new graphic objects and the manipulation of existing ones in many different ways. Extensive discussions on the subject are available in Baudin (2010), Lyshevski (2003), Kiusalaas (2005), among several others. In addition, extensive explanations, examples and other support are packaged with each software as well as the developers' websites and the user community. In the limited space available for the present discussion, attention

shall be focused on the generation of new graphical objects such as bar charts, pie charts, line graphs and surface plots.

8.4.1 Bar Charts

There are two functions for the generation of bar charts, namely **bar** and **barh**. The two functions are similar, except that the function **bar** draws vertical bar charts, while **barh** draws horizontal bar charts. Given a column vector **x** and a matrix **y** of one or more columns but the same number of rows, the command **bar(y)** produces a bar chart of **y** with the default specifications. The function may also be called in two other ways, namely

bar(y)
bar(x,y)
bar([h],x,y [,width [,color [,style]]])

The arguments are described as follows:

h an axes handle, (default: h=gca()).

y a real scalar, vector of size N, or a matrice N*M.

x a real scalar or a vector of size N, (default: if y is a vector then x is a vector and x length equals to y length. If y is a matrix then x is a vector with length equals to the number of rows of **y**.

width (optional), a real scalar, defines the width (a percentage of the available room) for the bar (default: 0.8, i.e 80%).

color (optional), a string (default: 'blue'), specifing the inside color bar.

style: a string, which should be 'grouped' or 'stacked' (default: 'grouped').

As an example, consider the data for variables **x** and **y** shown in table 8.3. Running the code shown produces the bar chart shown in figure 8.1.

As shown in line 13 of table 8.3, the legend was produced with the command **legend('Income','Expenditure','Surplus',"in_upper_left")**, while the chart was saved as a PNG file with the filename **barchartexample.png** using the export function **xls2png**.

We note that the number of graphic windows so far opened, **count**, is defined in an initialization file which must be run before any other script file. Similarly, the directory path, **direxample**, is defined in the initialization file.

Table 8.3: An example code to create a barchart

```
x = [2005. 2006. 2007. 2008. 2009. 2010. 2011]';
y =[13.  9.  ;
 21.  18.  ;
 15.  13.  ;
 17.  14.  ;
 20.  29.  ;
 32.  36.  ;
 15.  17];
count = count +1;//updatenumber of graphic windows so far opened
y(:,3) = y(:,1) - y(:,2);//determine the difference
bar(x,y);//create the barchart
xtitle('','Year','Income,Expenditure,Surplus');//define the x and
    y axis titles
legend('Income','Expenditure','Surplus","in_upper_left");//Define
    the legend and place in upper left corner of figure
myfile = 'barchartexample.png';//specify filename
myfilename = mydirexample + '/' + myfile;//define filename with
   full path specifications
xs2png(count,myfilename);//save the file in PNG format with the
    specified name
```

8.4.2 Pie charts

Drawing a pie chart is equally simple. Given a vector **y** consisting of **n** elements, the command **pie(y)** draws a pie chart with **n** sectors, each in proportion to the value of the element of **y**.

By default, the value of **y(i)** expressed as a percentage of the sum of the all the elements of the vector, **Y(i)** is printed against the sector, where

$$Y(i) = \frac{y(i)}{\sum_{i=1}^{n}} \times 100 \tag{8.1}$$

Each sector has a different colour. When $n > 7$, the colours are repeated in sets of 7.

Variations of the basic command are available to configure the pie chart.

For example, given the data **y = [2 5 9]**, the pie chart is drawn using the command **pie(y)**, resulting in the pie chart shown in figure 8.2.

More sophisticated formatting of the pie chart is possible. For example, using the code of table 8.4, the pie chart shown in figure 8.3 can be generated.

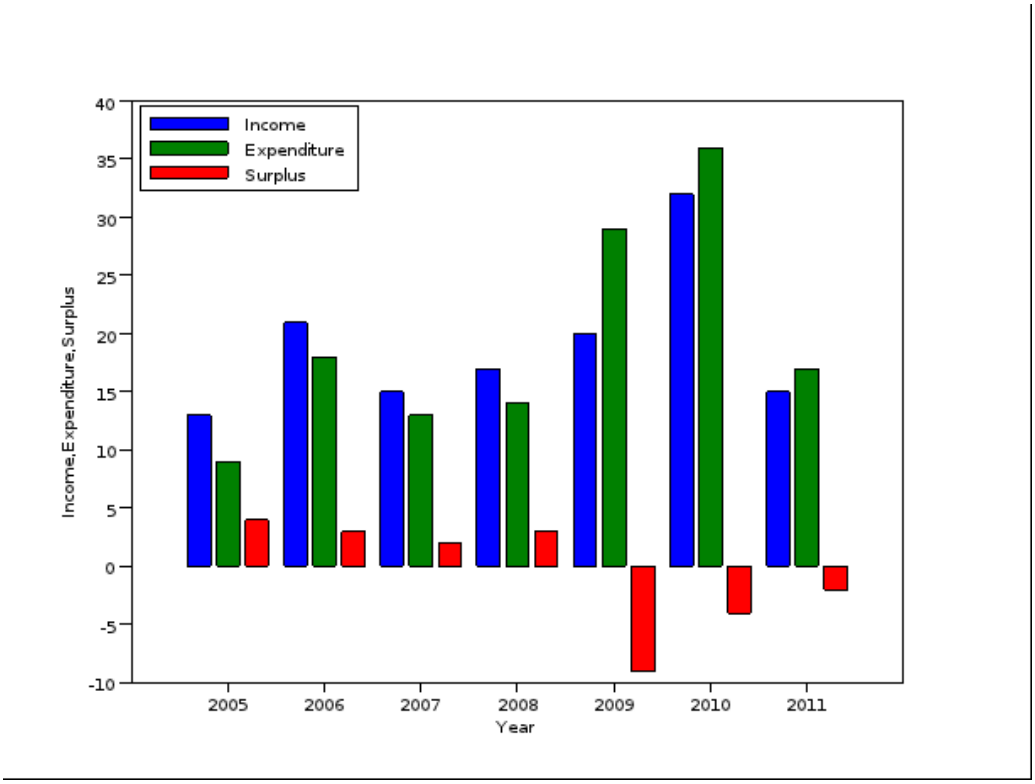


Figure 8.1: An example bar chart produced with code and data in table 8.3

8.4.3 2D plots

Plots of data and functions in 2 dimensions can be generated with the simple command **plot(y)**, where **y** is a vector or matrix of real numbers. The x-axis of the plot may be specified more explicitly using the command **plot(x, y)**. The title of the plot, together with the x and y-axis labels can be specified with the command **xtitle('title', 'x-axis label','y-axis label')** where the parameters in quotes should be modified by the user.

If there are several plots on the same graph, it is often better to distinguish them using different line or point styles. Thus, to indicate the data points with an asterisk (*), join them with solid lines and use a red color for the line and points, we add line specifications to the plot command such as **plot(x, y, 'r*-')**, where the characters 'r', '*', and '-' in the line specifications can each be replaced to obtain other effects. For example, the 'r' in the line specifications may be replaced by 'g', 'b' or 'm' respectively to produce green, blue or magenta colours. Similarly, the '*' in the line specifications may be replaced by 'd' or '+' to replace the astericks with diamond or '+' marks at the data positions. Other possible variations are indicated in the software documentation.

As an example, consider the number of hours of daylight for Abuja, Nsukka and Sokoto shown in

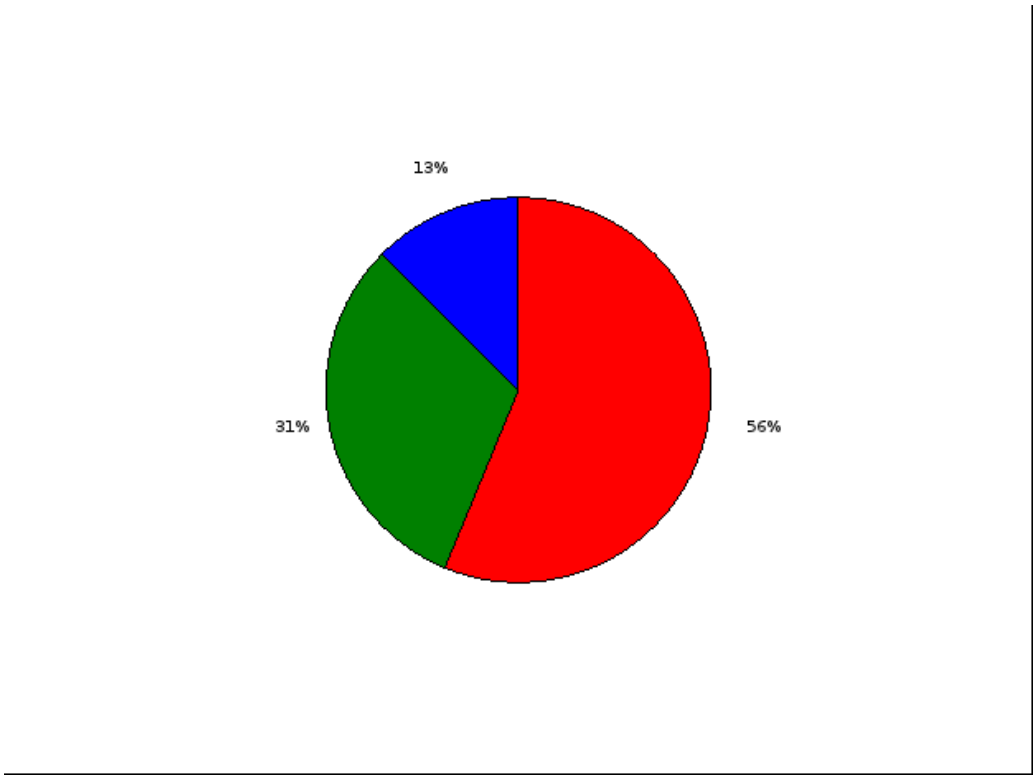


Figure 8.2: A simple pie chart

lines 2–4 of the programme code in table 8.5.

Executing the file in Scilab results in the graph in figure 8.4

The three plot commands in lines 7 to 9 may be combined into a single plot command by repeating the parameters in the three plot commands, as shown in line 17. The plot generated are exactly the same as before.

The user has to strike a balance between compactness and readability in deciding when to use a single or multiple plot command when there are several plots on the same graph. The x- and y-axis labels in the figure were produced with the xtitle command. As expected, the legend was produced by the command **legend ('Abuja', 'Nsukka', 'Sokoto')**.

Table 8.4: Code to create the piechart shown in figure 8.3

```
filename = mydirfig + '/yearest.csv';//specify name of csv file
    to read in dirfig
count = count + 1;//number of graphic objects so created
mytext = read_csv(filename);//read file contents and assign to
    string matrix zz
mytext(:,3) = '';//use null entries for column
yy = evstr(mytext);//convert to numerical values
wn = scf(count);//creat new graphic  window
pie(yy(:,2),mytext(:,3));//draw the pie chart without labels in
    place
legend(mytext(:,1));//place labels in a legend
myfilenamebase = 'piechart-example.png';
myfilename = mydirfig + '/' + myfilenamebase;//create filename
    with full path
xs2png(count,myfilename);//save plot as PNG file with specified
    filename
```

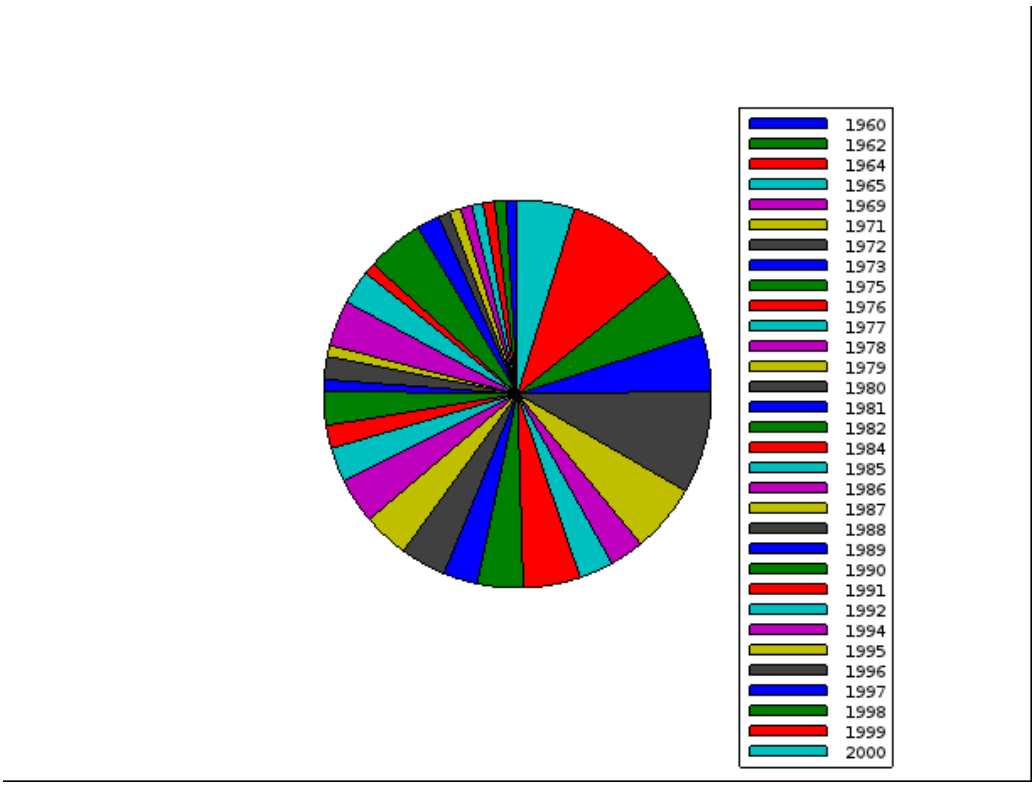


Figure 8.3: A pie chart with legend placed outside
(The SCILAB code is shown in table 8.4)

Table 8.5: Sample Scilab code for 2-d plots of data

```
month = 1:12;//months of the year
Abuja = [11.523 11.713 11.947 12.207 12.425 12.532
    12.484 12.298 12.048 11.789 11.573 11.469];
Nsukka = [11.641 11.784 11.96 12.156 12.319 12.4 12.364 12.224
    12.036 11.841 11.679 11.601];
Sokoto = [11.323 11.593 11.925 12.293 12.602 12.755 12.686 12.423
    12.068 11.701 11.394 11.247];
count = count + 1;
scf(count);//create new graphic window
plot(month, Abuja, 'r*-');
plot(month, Nsukka, 'gd-');
plot (month, Sokoto, 'b+-');
xtitle('','Month', 'Number of hours of daylight');
legend('Abuja','Nsukka','Sokoto');
myfilenamebase = 'plot-example.png';
myfilename = mydirfig + '/' + myfilenamebase;//create filename
    with full path
xs2png(count, myfilename);//save plot as PNG file with specified
    filename
count = count + 1;
scf(count);//create a new plot window
plot(month, Abuja, 'r*-',month, Nsukka, 'gd-',month, Sokoto, 'b+-
    ');
xtitle('','Month', 'Number of hours of daylight');
legend('Abuja','Nsukka','Sokoto');
```

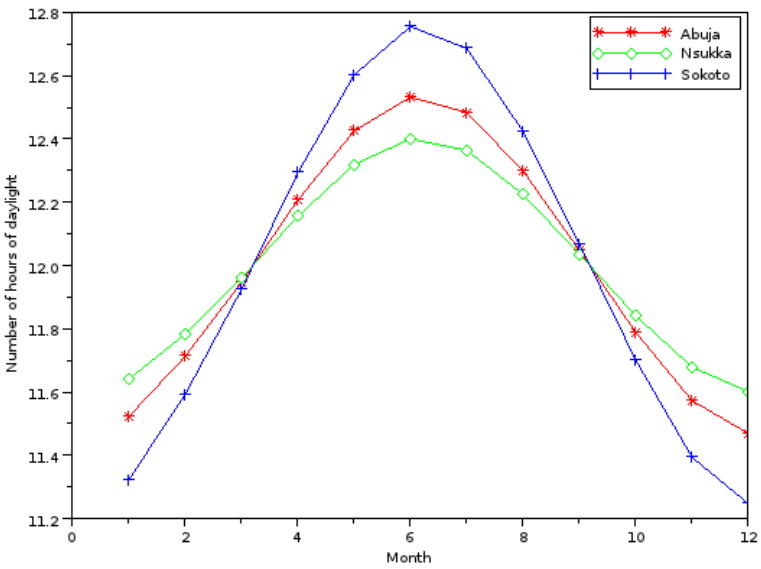


Figure 8.4: Number of hours of daylight at Abuja, Nsukka and Sokoto

8.4.4 3D and other plots

Several other plotting commands are also available. These include **plot3d** (for plotting in three-dimensions), **contour** (for preparation of contours), among others. Details may be found in the relevant user manuals.

As an example, the codes in table 8.6 produced the surfaces in figure 8.5.

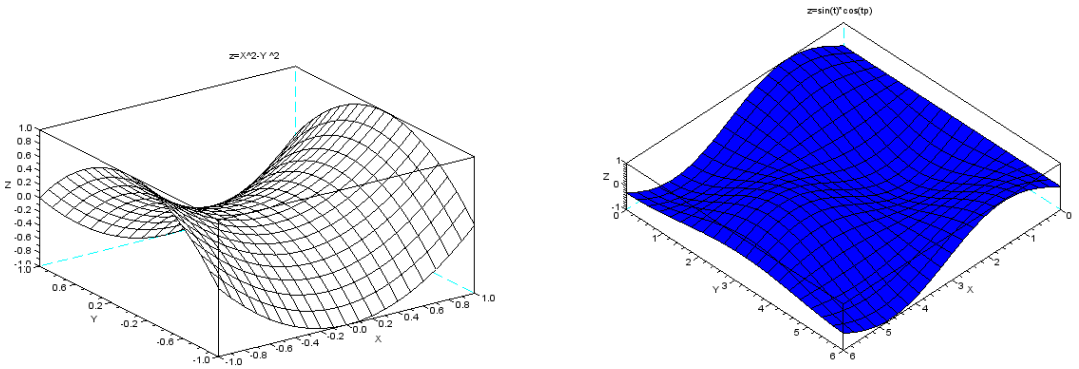
Table 8.6: Sample Scilab code for 3-D plots of funstions

```
count = count + 1;
n = scf(count);//create new graphic window
[X,Y]=meshgrid(-1:.1:1,-1:.1:1);
Z=X.^2-Y.^2;
xtitle('z=X^2-Y ^2');
mesh(X,Y,Z);
filenamebase = 'mesh-example.png';
filename = mydirfig + '\' + filenamebase;
xs2png(n, filename);//save file with filename specified

// simple plot using z=f(x,y)
count = count + 1;
n = scf(count);//create new graphic window
t=[0:0.3:2*%pi]';
tp = t';
z=sin(t)*cos(tp);
xtitle("z=sin(t)*cos(tp)");
plot3d(t,t,z)
filenamebase = 'plot3d-example.png';
filename = mydirfig + '\' + filenamebase;
xs2png(n, filename);//save file with filename specified
```

8.5 Input/output

MATLAB and SCILAB have very stable and reliable input/output algorithms and easy-to-use user interfaces to access them. Only the basic features will be highlighted in this paper.



(a) Produced with **mesh** function

(b) Produced with **plot3d** function

Figure 8.5: Illustratrations in 3D produced with the Scilab codes shown in table 8.6

8.5.1 Input

Input of data could be through the keyboard, mouse, data saved in a file, among others.

Keyboard input

Data input from the keyboard is obtained using the command **input**.

For example, the two expressions below will prompt the user for input.

```
x=input("Press any number");//prompt user for name input
x=input("What is your name?","string");//prompt user for name
```

For the first case, any data entered is evaluated and a number is returned as value of **x**, while a string is returned in the second case. The optional word,'string' in the second case may be shortened to simply 's'.

Spreadsheet

When data longer than a few words or numbers is required, it is better to organise them in a spreadsheet and save them in one or more files. The format of the data should be decided by the user. The data should be organized so as to make maximum use of the properties of matrices. In particular, the data should be arranged such that the first row is the header row which gives brief information about the content of each column. It is further recommended that:

- all the data in a given column (apart from the header row, if any) be of the same data type.

- If writing to a text file
  - Columns should be separated by comma, tab or space.
  - If columns are separated by the tabs or space character, missing data should not normally be permitted. When this is not possible, missing values should be represented by a unique character, number or string which is different from the regular data. For example, in matrices with numerical data, it is common to represent missing values with NaN (i.e. not a number).

To read a text file with comma separated values, the function **read_csv** should be used. The general calling sequence is

```
M = read_csv(filename, sep);//read comma separated value file
    with given separator
```

The filename should include the full path of the file, as well as the extension. If the second input argument, **sep** is ommitted, a tab separator is assumed. The variable **M** in the output field is a string matrix. To convert it to numerical values, we use the **evstr** function, and have, for example, **Num = evstr(M)**.

To read a spreadsheet file saved in the Microsoft Excell 1997 to 2003 format (.xls), the function **readxls** should be used. The general format of the function is

```
sheets = readxls(file_path)
```

where

file_path a character string: the path of the Microsoft Excel file.

sheets an mlist of type xls, with one field named sheets

Given an Excel file path, this function returns an mlist data structure of type **xls**, with one field named sheets. The sheets field itself contains a list of sheet data structure. Thus we have

```
sheet=mlist(['xlssheet','name','text','value'],sheetname,Text,
    Value)
```

where sheetname is a character string containing the name of the sheet, Text is a matrix of string which contains the cell's strings and Value is a matrix of numbers which contains the cell's values. The sample code in table 8.7 opens and reads the first sheet in an Excell file named **expenditure.xls** in the directory **mydirexample**. The text values are assigned to the variable **mytext** while the numerical values are assigned to the variable **myvalue**.

To generalize the above processes for any given sheet, we employ the user-defined function *readxlsfile* which is called using the following syntax:

Table 8.7: Sample Scilab code to open and read a Microsoft Excell file

```
myfilename = mydirexample + '/expenditure.xls';//define filename
Sheets = readxls(myfilename);//open and read Excell 2003 file
// some basic operations on Sheets
typeof(Sheets)//print the type of the sheets
s1=Sheets(1); //get the first sheet
typeof(s1)//print type of the first sheet
myvalue = s1.value; //get the first sheet value field
mytext = s1.text;  //get the first sheet text field
myheading = mytext(1,:);//name first row as heading
s1(2,:)  //get the 2nd row of the sheet
typeof(s1(2,:))
```

```
[myheading, mytext, myvalue] = readxlsfile (mydir, myfilename,
    sheetnum);//open an Excell file, read contents, extract header
    , and return the text and value portions of the data
```

Thus, the function requires two or three input variables and produces three output variables as follows:

Input varables There are two input variables, as follows:

mydir This is the full path of the directory where the file to be opened is located.

myfilename This is the name of the file (including the extention). Presently, only files with the *.xls* extension have been tested.

sheetnum This is the sheet number of the spreadsheet file.

Output variables The three output variables are explained as follows:

myheading This variable returns the first row of the data as the heading. It is assunmed to contain only strings.

mytext This variable contains all the text of the original file from the second to the last row.

myvalue This variable contains all the numerical data in the file from the second to the last row. Cells with missing values are usually represented with !Nan (not a number).

We note that the above function considers only the specified sheet in the input file. If the sheet number is not specified, then the first sheet is assumed. It can be generalized to handle all the sheets in the file. Readers interested in this aspect should consult the full online documentation of the function *readxls* in MATLAB or SCILAB.

Since the first row of the data is used as header, we remove that in the data before any further analysis. This is implemented within the function *readxlsfile* using the expressions,

```
heading = s1.text(1,:); //assign first row to a new variable
s1.text(1,:) = []; // delete first row of text
s1.value(1,:) = []; // delete first row of value
```

The variable *heading* preserves the header information for all the columns, and could always be viewed to check type of data in any column during the course of analysis. The original first row is deleted from the object s1 in order to simplify the code required for further analysis (see lines 2 and 3 above).

8.5.2 Output

The standard output for all variables, error messages, graphical objects, etc is the screen. Several functions are available to redirect any desired output to a file in a specified format. For example, to redirect output to a comma separated value file (.csv), the function **write_csv** may be used.

The general format for calling the function is

```
write_csv(M, filename [,sep, dec])
```

where

filename a character string. The file path.

M a matrix of strings.

sep column separator mark, by default a tabulation: ascii(9) or "\t ".

dec decimal mark '.' or ',' by default a ','

Three examples are shown in table 8.8. Here, data are prepared and saved to different text files with comma separated values using the **write_csv** function.

To redirect output to a file for inclusion as a LaTeX table, the function **writecsv** may be used, with the columns separated by the '&' character. The end of line indicators, '\}\verb}' should be added as the last column of the table before calling on the **write_csv** function. This is shown in line 24 of table 8.8

Table 8.8: Sample Scilab code to write data to a comma separated value file

```
// save a matrix as csv file format
A = [1:10]';//transpose of first ten numbers
A(:,2) = A(:,1) * 0.1;
A(:,3) = A(:,1).^2;
filename = mydirexample + '/writecsvexample1.csv';
write_csv(A,filename );

// read as text using the mgetl function
AA = mgetl(filename);//read all lines as text

r = read_csv(filename,ascii(9));
r = strsubst(r,',','.');
ra = evstr(r);//convert to numbers

filename2 = mydirexample + '/writecsvexample2.csv';
write_csv(A', filename2, ' ', '.');
B = mgetl(filename2);//read all lines as text
BB = evstr(B);//convert to numbers

//Saving file as LaTeX table
filename3 = mydirexample + '/writecsvexample3.tex';
AB = string(A);//convert to string
[m,n] = size(AB);//get dimensions
AB(:,n+1) = '\\\hline';//add LaTeX end of line characters
write_csv(AB,filename3,'&' );//separate columns with & character
```

8.6 Statistical Analysis

There are specific functions to perform most standard statistical analysis in SCILAB or MATLAB. In addition, there are toolboxes designed to handle specialised applications. For example, in SCILAB, the 55 basic statistical functions are listed in table 8.9.

Table 8.9: Basic statistical functions available in SCILAB

Function	Description
cdfbet	cumulative distribution function Beta distribution
cdfbin	cumulative distribution function Binomial distribution
cdfchi	cumulative distribution function chi-square distribution
cdfchn	cumulative distribution function non-central chi-square distribution
cdff	cumulative distribution function F distribution
cdffnc	cumulative distribution function non-central f-distribution
cdfgam	cumulative distribution function gamma distribution
cdfnbn	cumulative distribution function negative binomial distribution
cdfnor	cumulative distribution function normal distribution
cdfpoi	cumulative distribution function poisson distribution
cdft	cumulative distribution function Student's T distribution
center	center
wcenter	center and weight
cmoment	central moments of all orders
correl	correlation of two variables
covar	covariance of two variables
ftest	Fischer ratio
ftuneq	Fischer ratio for samples of unequal size.
geomean	geometric mean
harmean	harmonic mean
iqr	interquartile range
mad	mean absolute deviation
mean	mean (row mean, column mean) of vector/matrix entries
meanf	weighted mean of a vector or a matrix
median	median (row median, column median,...) of vector/matrix/array entries
moment	non central moments of all orders
msd	mean squared deviation

Table 8.9 continued: Basic statistical functions available in SCILAB

Function	Description
mvvacov	computes variance-covariance matrix
nancumsum	Thos function returns the cumulative sum of the values of a matrix
nand2mean	difference of the means of two independent samples
nanmax	max (ignoring Nan's)
nanmean	mean (ignoring Nan's)
nanmeanf	mean (ignoring Nan's) with a given frequency.
nanmedian	median of the values of a numerical vector or matrix
nanmin	min (ignoring Nan's)
nanstdev	standard deviation (ignoring the NANs).
nansum	Sum of values ignoring NAN's
nfreq	frequence of the values in a vector or matrix
pca	Computes principal components analysis with standardized variables
perctl	computation of percentils
princomp	Principal components analysis
quart	computation of quartiles
regress	regression coefficients of two variables
sample	Sampling with replacement
samplef	sample with replacement from a population and frequences of his values.
samwr	Sampling without replacement
show_pca	Visualization of principal components analysis results
st_deviation	standard deviation (row or column-wise) of vector/matrix entries
stdevf	standard deviation
strange	range
tabul	frequency of values of a matrix or vector
thrownan	Eliminates nan values
trimmean	trimmed mean of a vector or a matrix
variance	variance of the values of a vector or matrix
variancef	standard deviation of the values of a vector or matrix

Comparable functions are available in MATLAB, with similar names and user interface.

More specialized statistical functions are available as toolboxes. In SCILAB, these can be installed online via ATOMS. The statistical toolboxes available for download from the SCILAB website as at 21/11/2011 are listed in table 8.10.

Table 8.10: Statistical toolboxes available in SCILAB

Function	Description

The same tasks could also be performed in MATLAB using equivalent functions or toolboxes.

8.7 Recommended Procedure

In order to routinely employ MATLAB and/or SCILAB for data analysis, a multi-step procedure involving the following components is recommended:

- Data Generation
- Data Entry
- Data Cleaning
- Data Analysis/Plotting/Visualizsation
- Report Framework Generation
- Draft/final Report Preparation using the Report Framework

Each of these steps will now be discussed in some further detail.

8.7.1 Data Generation

The first and obvious step is the generation of the data for analysis. In designing the data collection instrument or procedure, it should be noted that the quality of data input will greatly affect the resulting data output and the relevance and usefulness of the results. Every effort should be made to apply the best practices in experimental design, as recommended in Chukwu (2011).

8.7.2 Data Entry

The next step is to enter the data to a computer file which will latter be read by MATLAB/SCILAB. This can be done in several ways such as writing the data to a text file or spreadsheet. Details are presented in section 8.5.1.

8.7.3 Data Cleaning

Data entry should be followed or accompanied by data cleaning so as to identify and remove errors due to mistakes in manual entry or from other sources. For small volumes of data, this could be achieved by printing the data and checking the entries manually, after which corrections are made in the computer file. When this is impractical because of large volumes of data involved, a function could be written to check individual entries electronically to remove data that are out of range for each column.

8.7.4 Data Analysis/Plotting/Visualization

Once the data entries have been properly cleaned, processes of data analysis/plotting/visualization could proceed. While the specific details will depend on the available data and the overall goal of the project, the following general procedure could be adopted:

- The available data are thoroughly examined and the different kinds of output data required are determined. From this, the types of analysis needed to convert the input data into the output data are obtained.
- A function is written to implement the actual analysis of each kind of data. As much as possible, the input and output parameters to each function should be matrix objects of the appropriate type. In this way, the function could become as general as possible and not tied to the specific problem being solved.
- In the same way, the different types of graphic and tabular displays needed to make a good presentation of the results are determined, and a function written to produce it from the output data.
- Each function is debugged and tested as it is being developed, using, if possible, the input data.

8.8 Report Framework Generation

It is recommended that a special function be written to generate the skeleton or framework of the project report. This will essentially be a file that lists each table or illustration generated for possible inclusion in the project report. For completeness, the tables and illustrations may be garnished with the appropriate captions and cross-referencing information. For users of the LaTeX document preparation systems, this is very easy and convenient. A simple user-defined function which may be employed in this, is *latexreport*.

The calling sequence is as follows:

fid = latexreport (fid, sectiondepth, sectiontext, closefile, directory, filenamebase, comment); // prepare draft report framework for the project in LaTeX.

Thus, the function has one output variable and seven (7) input variables. The variable *fid* is the file identification number for the file to which the report should be written. If the file is not already open, the function creates such a file and opens it. The other input variables are as follows:

sectiondepth This is an integer quantity between 1 and 4 to indicate the depth of the section as shown in the table 8.11.

Table 8.11: LaTeX sectioning depth values for use in in function *latexreport*

Value of section depth	1	2	3	4
LaTeX Sectioning Command	\section	\subsection	\subsubsection	\textbf

sectiontext This is the descriptive text which serves as the sectional heading.

closefile If this variable is greater than zero, the file is closed at the end of the call, otherwise it remains open for further entries.

directory This indicates the directory (with full path) of the report framework.

filenamebase This is the base name of the report framework file. The extension is assumed to be *.tex*

The function works in the same way in both MATLAB and SCILAB, and produces similar results.

8.8.1 Including Tables

Tables generated from the analysis or any other tabular material can easily be included in the report framework to speed up the task of writing the final report. For users of the LaTeX software, the function *latextable* can be employed. The general syntax for this function is as follows:

```
[fid,count] = latextable(fidreport, count, datatext, data,
   heading, dataformat, caption, dirtab, filenamebase, ext,
   mysum)//open a file and save contents of data and datatext
   with suitable formats for printing in \LaTeX\
```

Here, *fid* represents the index to the table saved on disk, while *count* indicates the number of tables so far created. The input parameters are explained as follows:

fidreport This is the file identification number for the report framework, as explained previously.

count Same as *count* in the output variable list.

datatext A matrix of strings identifying each row of the data to be printed

data The numerical data to be printed.

heading The first row of the table which serves as the heading.

dataformat The format for printing the data. For numerical data, the number of decimal places is indicated.

caption The table caption.

dirtab The directory where the table is to be saved.

filenamebase The filename for saving the table, exluding extension.

ext The filename extension, usually *.tex*.

mysum If greater than zero, data along each row are to be summed and the total included.

Alternatively, the data could be converted into string variables using the **string** function and then exported using the **write_csv** function as employed in line 22 of table 8.8.

8.8.2 Including Figures

All figures generated from the analysis or any other graphical material can easily be included in the report framework to speed up the task of writing the final report. For users of the LaTeX software, the function *latexfigure* can be employed. The general syntax for this function is as follows:

```
[count] = latexfigure(fidreport,count, directory, filenamebase,
   caption)//include figure in a \LaTeX\ document
```

The input parameters have the same significance as for the inclusion of tables.

8.8.3 Report Preparation

Using the skeleton or framework of the project report thus generated, textual explanations and discussion of the output data can now be added at the appropriate portions in order to produce the draft report. In turn, this draft project report can be edited and enriched with other useful information in order to produce the final report.

8.9 Examples

To illustrate some of the principles considered previously, a few examples covering some commonn problems will be presented.

8.9.1 Examination Results

As an example, consider the examination results of a group of students saved in spreadsheet file with the name *result.xls*.

With the continuous assessment and examination scores held in columns 5 and 6 respectively, the total mark is obtained from the expression

```
s1.value(:, 7) = s1.value (:, 5) + s1.value (:, 6)
```

To determine the individual letter grades, suppose that the grading system shown in table 8.12 is adopted for the present case.

Table 8.12: Example Grading System

Range of Scores	Letter Grade	Grade Point
$0 < 40$	F	0
$40 < 45$	E	1
$45 < 50$	D	2
$50 < 60$	C	3
$60 < 70$	B	4
$70 \leq 100$	A	5

To make the system somewhat more general, we save the above grading system in a spreadsheet file with the name *grading.xls*. The data is loaded into memory using the function *readxlsfile*. The letter grade is then calculated and assigned to column 8 of the text field by calling the user-defined function *lettergrade* as follows:

```
[ s1.text (:, 8), s1.value (:, 9)] = lettergrade (mytotal,
    myminscore, mylettergrade, mygradepoint);
mytotal = s1.value (:, 7); // column 7 just computed
myminscore = sgrade.value (:, 1);// min score for each grade
mylettergrade = sgrade.text (:, 3); // lettergrade
mygradepoint = sgrade.value (:, 4); // grade point associated
    with minimum score
```

The function *lettergrade* is employed for the detailed calculations. Internally, it employs the *for* loop to determine the first element of the vector *minimumscore* that a given item in the vector *total* is just equal to or greater than. The index of that score is noted, and becomes the index for the *lettergrade* and *gradepoint* objects.

For purposes of completeness, the output and input parameters to the function *lettergrade* are explained as follows:

Input Variables There are four input variables all of which are derived from table 8.12. These are as follows:

mytotal This is the column of total scores previosly computed.

myminscore This is the array of minimum scores for any particular grade. In the grading system of table 8.12, it corresponds to the numbers on the LHS of the inequality signs.

mylettergrade This is the array of lettergrades shown in column 2 of table 8.12

mygradepoint This is the array of numbers such that a given value is associated with the corresponding letter grade. For the system of table 8.12, it corresponds to the data of column 3.

Output Variables The output variables of this function are two, namely

lettergrade This is the array of letter grades associated with each candidate listed in the original input file.

gradepoints This is the array of gradepoints associated with the candidates in the input file.

Having computed the total score, grade and gradepoint for each candidate, the names are sorted in lexicographic order using the QuickSort algorithm as follows

```
s1 = gsort(s1, 'lr', 'i');//sort in lexicographical order
```

In calling the function gsort, the option 'lr' is selected to indicate lexicographic ordering of the rows, while the option 'i' ensures that it is in increasing order.

8.9.2 Counting of Strings

In the Social Sciences, a significant part of research is done with the use of questionnaire where respondents are expected to choose from among a given set of values (which may be numerical or textual).

Part of the analysis in such cases could be counting the number of times a particular choice is made and comparing this with overall matrix of choices made.

The analysis of such field data is probably better done with MATLAB/SCILAB than with other commonly used tools.

Consider for example the abridged field data shown in the spreadsheet file *fieldwork.xls*. This contains the responses of the staff of industrial firms in Nigeria to rate the performance of their production equipment using the scale

A. Very Efficient

B. Efficient

C. Not Efficient

Each firm was requested to indicate the rating of up to 10 different equipment using the same rating scale. The ratings were recorded separately under the headings *eqrate1* to *eqrate10*. To keep track of the individual firms, each is given a unique numerical code, and all the data for each firm are presented in the same row or in several consecutive rows. A sample of the first few lines of the data in a spreadsheet is shown in table 8.13.

Assuming that the order of the responses is immaterial (i.e rating for equipment listed as number 1 is equally treated with rating for any other equipment), then the total possible responses is obtained very easily. Using the function **readxlsfile** described previously, or otherwise, the spreadsheet file is opened and read using commands shown in lines 1 and 2 of table 8.14.

The number of responses corresponding to each letter is then counted using the function **tabul** as shown in line 16 of table 8.14.

The implementation of the above results in the output data shown in table 8.15.

The data may be illustrated with the piechart shown in figure 8.6.

8.9.3 General Mathematical Functions

There is full support for all of the standard mathematical functions, just as in C, C++, FORTRAN, among others. Some of these are shown in figures 13 and 14 on pages 22 to 23 of the **SCILAB Manual** (see Baudin (2010)) as well as several books on MATLAB (eg Lyshevski (2003)).. Consequently, the solution of problems involving any combination of standard mathematical functions is very easy.

For example, suppose we need to determine the exact times for sunrise and sunset as well as total number of hours of daylight for Abuja, Nsukka and Sokoto on the mean day of each month. The appropriate latitude and longitude of these towns are shown in table 8.16.

From Duffie and Beckman (1991), the hour angle at sunset, ω_s, is given by

$$\cos\omega_s = -\frac{\sin\phi\sin\delta}{\cos\phi\cos\delta} = -\tan\phi\tan\delta \tag{8.2}$$

Table 8.13: Sample of responses on rating of equipment

	A	B	C	D	E	F	G
1	codeno	eqrate1	eqrate2	eqrate3	eqrate4	eqrate5	eqrate6
2	306						
3	408						
4	711						
5	712						
6	713						
7	730	A	A	A	A	A	
8	304	B	C	C	C		C
9	510	B	B				
10	511						
11	514						
12	727	A	A	A	A	A	
13	103						
14	211	A	B	A	A	A	
15	224						
16	228						
17	702						
18	705						
19	232						
20	724	C	C	C	C		
21	405						
22	208	A	A	A	A	A	
23	302	A	A	A			
24	250	B	B	B	B	B	B
25	230	A	A	A	A	A	A
26	311	A		A	A		A
27	725						
28	401						
29	229						
30	502	A	A	A			
31	509						
32	721	A	A	A	B		
33	301	B	B	B	B	B	B
34	219						
35	221						
36	505	A	A	A	A	A	A
37	305	B	B	B	B	B	C
38	721	B	B	B	B	B	
39	231	B	B	B	B	B	B
40	513	A	A	A	A	A	A
41	226	C	B	B			

Table 8.14: An example code to count strings from a field survey

```
filename = mydirexample + '/fieldwork.xls';
Sheets = readxls(filename);//open and read Excell 2003 file
s1=Sheets(1); //get the first sheet
myvalue = s1.value; //get the first sheet value field
mytext = s1.text;  //get the first sheet text field
myheading = mytext(1,:);//name first row as heading
mytext(1,:) =[];//delete first row of text
myvalue(1,:) =[];//delete first row of value field
inputcol = [168 174 180 186 192 198 204 210 216 222];//input
   columns for rating of equipment
[m,n] = size(mytext);//determine number of rows and columns
mytextdata = [];//initialise textdata
for k = 1:length(inputcol)
     mytextdata(:,k) = mytext(:,inputcol(1,k));//populate textdata
end
mytextdata = convstr(mytextdata,"u");//convert to uppercase to
   ensure uniform comparison.
mycount = tabul(mytextdata, 'i');//count strings in increasing
   order
mycountfreq = mycount(2)(3:5);//select the last 3 elements of
   second object of mycount
mycountfreqsum = sum(mycountfreq);//obtain totals
mycountfreq(:,2) = mycountfreq(:,1) * 100/mycountfreqsum;//
   determine the percentages
```

while the number of daylight hours, is given by

$$N = \frac{2}{15}\cos^{-1}(-\tan\phi\tan\delta) \tag{8.3}$$

In these equations, ϕ is the latitude, while the declination angle, δ, is given by

$$\delta = 23.45\sin\left(360\frac{284+n}{365}\right) \tag{8.4}$$

The variable n is called the daynumber, where $n = 1$ on January 1, and $n = 365$ on December 31. The value of n for the i^{th} day of each month is shown in column 2 of table 8.17.

To implement these calculations in a very convenient and general manner, we define the function

Table 8.15: Rating/Performance of Equipment

Response	Rating by Firms	Number of Responses	Percent of Total
A	Very Efficient	242	50.31
B	Efficient	206	42.83
C	Not Efficient	33	6.86
–	Total	481	100.00

Source: Mbaya *et al*, (2011)

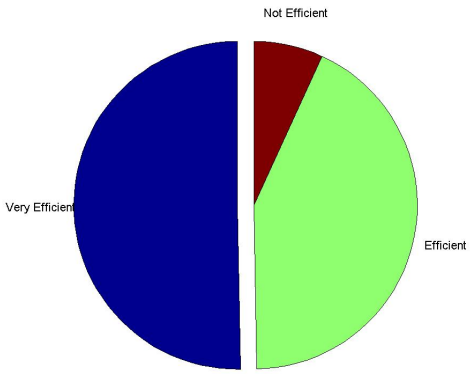


Figure 8.6: Pie chart illustrating the responses on equipment rating
Source: Mbaya *et al*, (2011)

Table 8.16: Approximate latitude and longitude (in degrees) for some towns in Nigeria

Location	Abuja	Nsukka	Sokoto
Latitude	9.27	7	13.03
Longitude	7.03	7	5.27

Table 8.17: Recommended Average Days for each Month and Values of n by Months

		For the Average Day of the Month		
Month	n for i^{th} day of Month	Date	n, Day of Year	δ, declination
January	i	17	17	-20.9
February	31+i	16	47	-13
March	59 +i	16	75	-2.4
April	90 + i	15	105	9.4
May	120 + i	15	135	18.8
June	151 + i	11	162	23.1
July	181 + i	17	198	21.2
August	212 + i	16	228	13.5
September	243 + i	15	258	2.2
October	273 + i	15	288	-9.6
November	304 + i	14	318	-18.9
December	334 + i	10	344	-23

Source: Duffie & Beckman, 1991

daynumber to determine the value of n for any given day of any given month, as shown in the file of the same name. The calling syntax is

> *mydaynumber = daynumber(month, day);//determine the daynumber for a given month and day*

Similarly, the declination, angle δ, is calculated using the function *declination*, as shown in the electronic file of the same name. The calling syntax is

> *[decrad, decdeg] = declination(daynumber) //This function is used to calculate the declination angle for any given day in degrees and radians. The input data is daynumber, where Jan 1 = 1, and Dec 31 = 365*

Here, *decrad* is the declination angle in radians, while *decdeg* is the same angle in degrees.
In the same way, the sunset hour angle, ω_s, is calculated both in degress and radians using the function *sunsethourangle*. The calling syntax is

> *[wsrad, wsdeg] = sunsethourangle(latitude, declinationangle, degrad);//*

where *wsrad* and *wsdeg* are the sunset hour angles in radians and degrees respectively. The parameter *degrad* is a boolian variable which can be zero (0) or 1. If degrad = 1, then input angles are in degree, otherwise they are in radians.

Finally, the number of daylight hours in the day may be determined using the function *daylength* or *daylengthd*. The second function is used if calculations in degrees are preferred. The calling syntax is

daylighthours = daylength(mysunsethourangle);//determine the number of hour of sunlight for the day

Having defined all the relevant functions, the calculations for the three towns are performed using the SCILAB code shown in table 8.19

The implementation of the code in table 8.19 results in the data shown in table 8.18.

Table 8.18: Daylength and sunset hour angle for selected Nigerian cities

	Daylength			Sunset Hour Angle		
	Abuja	Nsukka	Sokoto	Abuja	Nsukka	Sokoto
Month	Hours	Hours	Hours	Deg	Deg	Deg
JANUARY	11.523	11.641	11.323	86.423	87.310	84.926
FEBRUARY	11.713	11.784	11.593	87.848	88.381	86.948
MARCH	11.947	11.960	11.925	89.605	89.703	89.440
APRIL	12.207	12.156	12.293	91.551	91.167	92.199
MAY	12.425	12.319	12.602	93.184	92.395	94.516
JUNE	12.532	12.400	12.755	93.989	93.000	95.661
JULY	12.484	12.364	12.686	93.627	92.727	95.146
AUGUST	12.298	12.224	12.423	92.238	91.683	93.174
SEPTEMBER	12.048	12.036	12.068	90.362	90.272	90.513
OCTOBER	11.789	11.841	11.701	88.418	88.810	87.757
NOVEMBER	11.573	11.679	11.394	86.794	87.589	85.452
DECEMBER	11.469	11.601	11.247	86.018	87.005	84.349

The data for columns 2, 3 and 4 of the table are futher illustrated in figure 8.4 on page 111.

Table 8.19: Code to calculate the daylight hours

```
//determination of number of hours of daylight at Abuja,
      //Nsukka and Sokoto for the mean day of each month
town = {'Abuja', 'Nsukka','Sokoto'};
latitude = [9.27, 7, 13.03];
day = [17 16 16 15 15 11 17 16 15 15 14 10];
monthname = ['JANUARY' 'FEBRUARY' 'MARCH' 'APRIL' 'MAY' 'JUNE'
    'JULY' 'AUGUST' 'SEPTEMBER' 'OCTOBER' 'NOVEMBER' 'DECEMBER']';//names of
       the months transposed
monthdata = [1:12]';//serial numbers of the moths
for month = 1:12//take each month of the year
    myday = day(month);//use mean day of each month
    mydaynumber = daynumber(month, myday);
    [decrad, decdeg] = declination(mydaynumber);
    for mytown = 1:3
        mylatitude = latitude(mytown);
        [wsrad(month, mytown), wsdeg(month, mytown)] = sunsethourangled(
           mylatitude, decdeg);
        mydaylength(month, mytown) = daylengthd(wsdeg(month, mytown));
    end
end
daylengthwsdeg = [mydaylength, wsdeg];
      //combine the data into a single object to facilitate printing
heading = {'Month','Hours','Hours','Hours','Deg' ,'Deg','Deg'};
//Heading for the combined object
mycaption = 'Daylength and sunset hour angle for selected Nigerian cities';
       //descriptive caption for the table
dataformat = {'\%s','\%6.3f','\%6.3f','\%6.3f','\%6.3f','\%6.3f','\%6.3f'};
//format for printing every column of the table

filenamebase = 'daylength';//save table with this filenamebase
[fid, tablecount] = latextable(fidreport, tablecount,monthname,
        daylengthwsdeg,heading, dataformat, mycaption,
        dirtab,filenamebase,texextension,mysum0);
      //open a file and save contents of data and datatext with suitable
         formats for printing in LaTeX.
      //Use myfilename as label
x1 = monthdata;
y1 = mydaylength;
xlabeltext = 'Months';
ylabeltext = 'Number of Daylight Hours';
specification = '*-';//use * to indicate the points and join with lines
mylegendtext = ['Abuja','Nsukka','Sokoto'];//
countfigure  = createfigure(x1, y1, xlabeltext, ylabeltext,dirfig,
      filenamebase,specification,fidreport, mycaption, mylegendtext,
         countfigure)
```

References

Baudin M. (2010) Introduction to SCILAB. The SCILAB Consortium–Digiteo. Available online at `http://www.scilab.org`

Baudin M. Couvert V. and Serge Steer (2010) INRIA Paris - Rocquencourt. The Scilab Consortium–Digiteo / INRIA. Available online at `http://www.scilab.org`

Chukwu W. I. E. (2011 Design of Experiments for Efficient Data Analysis/Visualization. Chapter 8 in Enibe (2011) (ed) MATLAB/SCILAB for Data Analysis/Visualization. National Centre for Equipment Maintenance & Development, University of Nigeria, Nsukka

Duffie J. A. and Beckman W. A. (1991) Solar Engineering of Thermal Processes. Wiley, New York

Kiusalaas J (2005 Numerical Methods in Engineering with MATLAB. Cambridge University Press, Cambridge

Lyshevski S. E. (2003) Engineering and Scientific Computations using MATLAB. Wiley-Interscience, New Jersey

E. I. Mbaya, S. O. Enibe, & Engr. G. Ladan, (2011) INDUSTRIAL STUDIES On BASE METAL, IRON AND STEEL AND ENGINEERING SERVICES SECTOR (6^{TH} UPDATE(Draft)), Raw Materials Research and Development Council (RMRDC), Abuja

Stephen L. Campbell, Jean-Philippe Chancelier and Ramine Nikoukhah(2010) Modeling and Simulation in Scilab/Scicos with ScicosLab 4.4. Springer Science+Business Media, New York

INTRODUCTION TO SCILAB

Begining at the next page is the official documentation containing the concise and helpful introduction to the **SCILAB** software. The latest version can be downloaded freely from the SCILAB website `http://forge.scilab.org/index.php/p/docintrotoscilab/`